Energy, Mines and Resources Canada | Énergie, Mines et Ressources Canada

CANMET

Canada Centre for Mineral and Energy Technology | Centre canadien de la technologie des minéraux et de l'énergie

DIGITIZING PROGRAMS TO GENERATE INPUT DATA FOR A HAULAGE SIMULATION MODEL BY USING FITTING CUBIC CURVES

C. Neumiller, N. Stuart, M. Heximer, W. Veldhuis

November 1986

This document is an unedited interim report prepared for the Mineral and Energy Technology (CANMET)

Ce document est un rapport préparé et rédigé [illegible] l'Énergie (CANMET)

84 pp

ENERGY RESEARCH PROGRAM
COAL RESEARCH LABORATORIES
DIVISION REPORT ERP/CRL 86-129(TR)

MICROMEDIA

DIGITIZING PROGRAMS TO GENERATE INPUT DATA FOR A HAULAGE SIMULATION MODEL BY USING FITTING CUBIC CURVES.

by

C.NEUMILLER,* N.STUART**, M.HEXIMER*, V.VELDHUIS*

ABSTRACT

In this report, a set of programs used to digitize and plot mine haulage roads, is described. The method employed in the running of these programs is detailed, as are calculations and subroutines that are used. The structure of input/output files, graphs, and charts produced are also described.

The developed digitizing program will be used to enter data to the "Haulage simulation" model, presently under development by Surface Mining Laboratory, Operations Research group in Devon and the National Research Council in Ottawa.

---

* Co-op students, University of Alberta, Edmonton, Alberta

** Research Scientist, CANMET, CRL, Surface Mining Laboratory, Devon, Alb.

PROGRAMMES NUMÉRIQUES SERVANT À PRODUIRE DES DONNÉES D'ENTRÉE POUR UN MODÈLE DE TRANSPORT SIMULÉ, À L'AIDE DE COURBES CUBIQUES AJUSTÉES

par

C. Neumiller*, N. Stuart**, M. Heximer* et W. Veldhuis*

RÉSUMÉ

Le présent rapport décrit un ensemble de programmes utilisés afin de convertir en numérique et de tracer des courbes à partir de données sur les voies de roulage d'une mine. Les détails de la méthode employée pour exécuter ces programmes sont aussi présentés ainsi que les calculs et les sous-programmes utilisés. Le rapport contient de plus, une description de la structure des fichiers d'entrée et de sortie, des graphiques et des diagrammes produits au cours des travaux.

Le programme numérique que l'on a mis au point servira à l'entrée de données dans le modèle de transport simulé "Haulage Simulation Model". La préparation de ce modèle est effectué actuellement par le Groupe de recherches opérationnelles à Devon et par le Conseil national de recherches du Canada à Ottawa, au Laboratoire d'exploitation minière en surface.

---

*Étudiant en stage coopératif, Université de l'Alberta, Edmonton, Alberta.

**Chercheur scientifique, CANMET, LRC, Laboratoire d'exploitation minière en surface, Devon, Alberta.

TABLE OF CONTENTS

PAGE

ABSTRACT i
RESUME iii
TABLE OF CONTENTS v

1. INTRODUCTION 1

2. PROGRAM I: THE DIGITIZING PROGRAM 2
   2.1 The User Interface 2
   2.2 Output File Structure 5
   2.3 Subroutines Used 6
   2.4 Other Comments 7

3. PROGRAM II: THE CUBIC SPLINE/PLAN VIEW PROGRAM. 7
   3.1 The User Interface 8
   3.2 The Input File. 8
   3.3 The Output File 9
   3.4 Subroutines Used 10
   3.5 Graphs Produced 11

4. PROGRAM III: THE CROSS-SECTION PROGRAM 12
   4.1 The User Interface 12
   4.2 The Input File. 13
   4.3 The Output File 13
   4.4 Subroutines Used. 14
   4.5 Graphs Produced 15
   4.6 Other Comments. 16

5. SUBROUTINES: USER-WRITTEN SUBPROGRAMS 16
   5.1 The Charcount Subroutine 17
   5.2 The Myscale Subroutine. 17
   5.3 The Dataplt Subroutine. 18
   5.4 The Grade Subroutine. 18
   5.5 The Angle Subroutine. 19
   5.6 The Transform Subroutine. 25

5.7 The Angslope Subroutine 27
5.8 The Spline Subroutine 27
5.9 The Segment Subroutine 28
5.10 The Cugraph Subroutine 28
5.11 The Chart Subroutine. 29

6. CONCLUSION 30

7. RECOMMENDATIONS 30

8. APPENDICIES. 32

A.1 APPENDIX: PROGRAM LISTINGS 33
1. Digit. 34
2. Digcuspdis 38
3. Cross 43
A.2 APPENDIX: INPUT/OUTPUT DATA FILE LISTINGS. 49
1. Digit Output 50
2. Digcuspdis Output. 53
A.3 APPENDIX: SUBROUTINE LISTINGS. 56
1. Scale for File 57
2. Cugraph for File 58
3. Grade for File 59
4. Angle for File 60
5. Transform for File 63
6. Spline for File. 67
7. Segment for File 68
8. Chart for File 69
A.4 APPENDIX: GRAPHS 71
1. Digcuspdis Graphs. 72
2. Cross Charts 75

LIST OF FIGURES

PAGE

1. Sequence of Header Data..... 3
2. Angle Subroutine's Basis..... 21
3. Calculating Alpha: Case 1..... 22
4. Calculating Alpha: Case 2..... 23
5. Calculating Alpha: Case 3..... 24

1. INTRODUCTION

Numerous computer programs are available to predict the time taken for a mine truck to move between locations. The greater part of the input data required for such programs concerns the description of the route. This report describes a program which allows for mine roadways to be digitized as a number of discrete points in XY and or XYZ cartesian space (the set of points for which elevation is known is generally much smaller than the total number specified by N-S/E-W co-ordinates).

The first program in the suite allows for a data file to be created, representing a haulage road by digitizing points on a mine plan, and entering, where necessary, elevation, speed restriction changes and rolling resistance changes.

A second program takes the set of XY co-ordinates and fits a reasonable curve (cubic spline) through them. This allows for:

1) a realistic looking plot to be made of the road, and
2) the roadway to be divided into sections of known length.

A third program splines the vertical data, so as to produce a cross section of the road. Speed limits and rolling resistance are also given on the plot.

Finally, a fourth program utilizes the features of second and third programs to generate an output file which describes the roadway in detail, e.g. length, grade, speed restriction. This program would provide input to a haulage simulation program, presently under development by Operations Research group of the Surface Mining Laboratory.

All programs have been written in FORTRAN 77 and are installed at the Digital's VAX-750 computer.

2. PROGRAM I: THE DIGITIZING PROGRAM

This program is designed to digitize data in graphics form, then send it to an output file. Firstly, the header data are requested, then line data. No calculations are made in the program itself. The primary concern of the program is to format the output file in such a way that it can be used in a later program. The listing of the program is found in Appendix A1.

2.1. THE USER INTERFACE

This program may be activated once the STUDENT account, found in the Surface Mining Lab's VAX computer, has been accessed through the terminal plugged into the Calcomp digitizer, by typing:

"Run [IAN.CAROL DIGIT"

The program then prompts for a filename and header data for the output file, all of which is entered through the keyboard or with the digitizer keys. The header data includes:

1) origin,
2) "x" maximum and minimum,
3) "y" maximum and minimum,
4) north arrow, and
5) variable units.

An example of this can be seen in Figure 1.

The data for each line is then entered. The first and last points must have entries for all three of the additional data types, as the values for these points cannot be interpolated. When a new line is digitized, the 'A' button must be used to digitize the first point. A prompt for the linename then appears:

Enter a data file name

type **input**

Digitize the origin

hit '**0**' button over origin

Digitize the x-maximum

hit '**9**' button at a known maximum

Enter x scale values, minimum and maximum

type **first** x value, **maximum** x value

Digitize the y-maximum

hit '**8**' button at a known maximum

Enter y scale values, minimum and maximum

type **first** y value, **maximum** y value

Digitize 2 points to form the north arrow

hit '**1**' button over bottom point of north arrow

hit '**1**' button over top point of north arrow

Enter plot units

type **metres**

Enter elevation units

type **metres**

Enter rolling resistance units

type **per cent**

Enter speed restriction units

type **kmh**

Fig. 1 - Sequence of Header Data

2. PROGRAM I: THE DIGITIZING PROGRAM

This program is designed to digitize data in graphics form, then send it to an output file. Firstly, the header data are requested, then line data. No calculations are made in the program itself. The primary concern of the program is to format the output file in such a way that it can be used in a later program. The listing of the program is found in Appendix A1.

2.1. THE USER INTERFACE

This program may be activated once the STUDENT account, found in the Surface Mining Lab's VAX computer, has been accessed through the terminal plugged into the Calcomp digitizer, by typing:

"Run [IAN.CAROL DIGIT"

The program then prompts for a filename and header data for the output file, all of which is entered through the keyboard or with the digitizer keys. The header data includes:

1) origin,
2) "x" maximum and minimum,
3) "y" maximum and minimum,
4) north arrow, and
5) variable units.

An example of this can be seen in Figure 1.

The data for each line is then entered. The first and last points must have entries for all three of the additional data types, as the values for these points cannot be interpolated. When a new line is digitized, the 'A' button must be used to digitize the first point. A prompt for the linename then appears:

Enter a data file name

type **input**

Digitize the origin

hit '**0**' button over origin

Digitize the x-maximum

hit '**9**' button at a known maximum

Enter x scale values, minimum and maximum

type **first** x value, **maximum** x value

Digitize the y-maximum

hit '**8**' button at a known maximum

Enter y scale values, minimum and maximum

type **first** y value, **maximum** y value

Digitize 2 points to form the north arrow

hit '**1**' button over bottom point of north arrow

hit '**1**' button over top point of north arrow

Enter plot units

type **metres**

Enter elevation units

type **metres**

Enter rolling resistance units

type **per cent**

Enter speed restriction units

type **kmh**

Fig. 1 - Sequence of Header Data

Start a line,
hit 'A' button over first point of the line,
enter the line name, and
type 'RIDGE-HAUL'.

Once the linename has been entered, the following query appears on the screen:

Enter additional data:
'A' for rolling resistance,
'B' for elevation,
'C' for speed restrictions, and
'D' to go to the next point.

If 'A' is entered, rolling resistance data is entered, and the query is then repeated with the 'A' option removed. When 'B' is typed, elevation data is entered. The repeated query no longer has the 'A' and 'B' options available. The choosing of the 'C' option produces a different query, once the speed restriction data has been entered. The query looks as follows:

Type 'D' to go to next point, type 'E' to end the line, type 'F' to end the file

When 'A', 'B' or 'C' are chosen, the data must be entered in the following format. Any line of data that is to be followed by another, at the same point, must start with an ampersand. If no more data is to follow the line typed in, the line must start with the per cent symbol.

e.g. Type & RR 9.01
& EL 1520
% SR 50.5

A 'D' can be chosen at any time when the query appears as long as it is preceeded by a data line that contains the '%' or if the 'B' button is pressed. The 'B' button results in the recording of the point, but no other data is entered. The use of any other button will result in the query being

posted on the screen. Before the 'E' option can be used (to end a line), the 'D' button must be activated to record the last point of a line. To end the file, the '$' key is pressed, data entered for all three categories and the 'F' option is chosen. This closes the data file and stops the program.

2.2. OUTPUT FILE STRUCTURE

The output file consists of header data and line data. The header data includes the filename, x-value range, y-value range, north arrow's points 1 and 2, plot units, elevation units, rolling resistance units and speed restriction units. If later use warrants it, the plot units can be changed to [x] units and [y] units.

The line data includes the line name, each point in the line, and the additional data for each point, if any is given. The line name is preceded by a blank line to separate each line from the other and to separate the first line from the header data.

Each point must have x,y co-ordinates. When no other data is available, the co-ordinates are preceded by a 'B'. The first point of a line is started by an 'A', the last by a 'D'. The last point of the last line in the file begins with an '$'. If more than one line of additional data is added, any line with a data line following it is preceded with anampersand. If there is only one line of data or if it is the last line of data for a point, the line starts with a '%'.

e.g. Ridge-haul

```
A 0.00     0.00
& RR       9.12
& EL    1520.0
% SR       50.5
```

A listing of the output file appears in Appendix A2.

2.3. SUBROUTINE USED

The **DIGIT** program uses two user-written subroutines, found at the end of the program; **CALC** and **SCALE**. The **SCALE** subroutine finds a scaling factor which is used by the **CALC** subroutine to change the digitized values. The digitizer records points in [mm] to fit within the value range given. The factor is found by subtracting the initial scale value entered, '**start**', from the final scale value entered, '**maximum**', and dividing the result by the maximum value that was digitized:

factor = [**maximum** - **start**] / [**maximum**] ..... (1)

The **CALC** subroutine performs four tasks:

1) Moves the origin from digitizer units to (0,0), and
2) Subtracts the origin, in digitizer units, from the succeeding points.
3) Scales the points to fit within the value range given at the beginning of the program, using the factor found by the scale subroutine.

e.g. x = [x(terminal) * factor] + start ..... (2)

4) Truncates the button variable (characters that begin each line returned to the terminal from the digitizer) from format '**AD**' to '**A**', the format sent to the output file.

The two subroutines are found at the end of the program listing in Appendix A1.

2.4. OTHER COMMENTS

The button used to digitize the line points is very important. The button character is used, both by this program and by the subsequent cubic spline program to determine the calculations and manipulations that are to be performed on each point.

The digitizer reads point data in absolute terms, relative to the bottom left hand corner of the table surface as (0,0). The digitizer uses [mm] units.

Additional data must be entered in the order in which it appears in the query. There is no path to get back to a data type that was listed ahead of the data type entered. For example, if elevation has been entered, rolling resistance data cannot be entered for that same point.

3. PROGRAM II: THE CUBIC SPLINE / PLAN VIEW PROGRAM

The purpose of the 'DIGCUSPDIS' program is to visually check the input data. It takes the data produced by the digitizer program and plots it out, in plan view, on the Calcomp plotter. It also calculates the distance between each point. The output file produced includes header data and line data. The line data is split into additional data (rolling resistance, elevation, speed restrictions) and the distance between the points.

The header data are read first, then the line data. The line data is sorted into arrays, the points are plotted, as well as the north direction arrow (for the first line only). The program then begins the calculations, most of which are performed by subroutines. The grade at each point is calculated. The rest of the calculations are done for two points at a time. The first two points are transformed so that the initial point obtains the co-ordinates (0,0). The grade for each point is likewise adjusted. The

cubic spline equation constants are then calculated. The line between the two points is divided into segments, having its value calculated for the end point. The distance between the points is figured out. Next, all of the values, including the original points, are transformed back to the original axis co-ordinates. The curve produced is plotted. This procedure is repeated for every point except the last one of the line. Once all of the lines have been so reproduced, the axes and titles of the graph are plotted.

The listing of this program is in Appendix A1.

3.1. THE USER INTERFACE

The program, once the 'STUDENT' account has been accessed, is started by typing: "Run [IAN.CAROL]DIGCUSPDIS" on the terminal that is connected to the Calcomp plotter. No inputs are made while the program is running. However, before a run is made, certain factors must be considered. The file named as the input file in the beginning of the program must match the output file produced by the digit program. Also, the input required for the Calcomp subroutines must by supplied, particularily the axes lengths.

3.2. THE INPUT FILE

The input file used is the output file generated by the 'DIGIT' program. Some of the header data is used to produce the graph: the 'FILENAME' and 'LINENAME' become the graph title, the [x,y] value ranges provide scaling information and axis values, the north arrow is plotted to indicate the direction of the data, the plot units become part of the axes titles. Any additional data and their corresponding units are not used by this program; they are written to the output file.

The point data are used to plot the curves produced after calculations have been performed. The alphanumeric character at the beginning of the line

indicates whether the point is the beginning of a line, the end of a line, the last point in the file, or has additional data following it. The input file format is explained in Section 2.2; the listing for the file is found in Appendix A2.

3.3. THE OUTPUT FILE

The output file has two categories; the header data and the line data. The line data is further divided into two sections; the additional data section and the distance section. The header information includes the file name, the distance units, elevation units, rolling resistance units and the speed restriction units. The line data begins with the '**LINENAME**'. The '**DATA**' section is the first section listed. This section is headed by the title '**DATA**', then the data lines. The data lines include the point number, the array (data type, e.g. elevation array **EL**) and the corresponding data. This section is ended by a specific data line "**000 FF 0.0**".

```
e.g. DATA
       1 RR       9.12
       1 EL    1520
       1 SR       50
       000 FF 0.0
```

The distance section is headed by the title '**DISTANCE**'. This is followed by the point number minus one (**PN-1**) and the distance between the point and the last point. This section is ended by a specific data line as well "**000 0.0**".

```
e.g. DISTANCE
       1    6.15
       2   10.33
       3    4.01
       000 0.0
```

The listing of this output file is located in Appendix A2.

3.4. SUBROUTINES USED

Both user-written subroutines and Calcomp software are used by this program. The first user-defined subroutine is '**CHARCOUNT**'. This subprogram takes a character string of unknown length and counts the number of characters in it. The next subroutine is '**MYSCALE**'; it scales the data to fit evenly onto an axis. Both '**CHARCOUNT**' and '**MYSCALE**' are found in the '**SCALE.FOR**' file. The '**DATAPLT**' subroutine is found in the '**CUGRAPH.FOR**' file with the '**CUGRAPH**' subroutine. The former plots each data point, while the latter draws the axis and the graph titles. The grade and angle subroutines, found in the '**GRADE.FOR**' file, calculate the grade at each point. The '**TRANSFORM**' subroutine transforms points to new co-ordinates and can calculate the angle of the axis rotation. This subroutine shares the '**TRANSFORM.FOR**' file with the '**ANGSLOPE**', '**THETAA**', '**BETAA**', and '**GAMMAA**' subroutines. '**ANGSLOPE**' calculates the angles corresponding to the grades calculated and transforms them to the new axis alignment. '**THETAA**', '**BETAA**', and '**GAMMAA**' calculate angles; '**THETAA**' finds the angle of axis rotation, '**BETAA**' finds the angle the hypotenuse between two points makes with the axis, and '**GAMMAA**' the net rotation ('**BETAA-THETAA**'). The '**SPLINE**' subroutine calculates the cubic spline equation constants. The '**SEGMENT**' subroutine, the last user-written subroutine used in this program, determines the number of segments the line between two given points is divided into.

The Calcomp software used includes the following subroutines:

PLOTS: to initialize the plotter,
PLOT: moves the pen to a spot a given distance from the origin (in terms of x and y),
LINE: connects every point in the array with a straight line,
SYMBOL: plots titles, and
NEWPEN: changes plotter pens.

None of the subroutines used are found at the end of the program. For this reason, a more comprehensive description of the user-vritten subroutines mentioned above can be found in Section 5. Calcomp subroutine descriptions are located in the manual **"PROGRAMMING CALCOMP ELECTROMECHANICAL PLOTTERS"**.

3.5. GRAPHS PRODUCED

The graphs produced by the **'DIGCUSPDIS'** program are a plan view of the mine roads digitized. If the digitizing has been done correctly, the curves produced will look like the curves digitized.

Each curve is drawn on the same plot. If no more than four curves are to be plotted, each will be a different color; after four, the color sequence is repeated. None the less, each curve will have a different symbol as long as there are fever curves than characters in the character set used (in this case, the ASCII character set is used).

Every plot has the following elements:

1) [x] and [y] axes, each with titles and scale annotations,
2) a graph title containing the filename and linename(s),
3) north direction arrow ('N' is drawn beside it), and
4) curves with symbols plotted at every recorded point.

The graphs produced by the **'DIGCUSPDIS'** program are presented in Appendix A4. The original drawings from which the digit program produced the input data files for the **'DIGCUSPDIS'** program are found in Appendix A4.

4. PROGRAM III: THE CROSS-SECTION PROGRAM

The purpose of this program is to plot the elevation (cross-section), rolling resistance and speed restriction profiles of a digitized mine road. Header data from the input file is read first, then the elevation, rolling resistance and speed restriction data is read. Any interpolation that needs to be done is performed at this time.

The distance between each point is read next. A running total is kept of the distance as each line is read; the resulting value is assigned as the distance of the point from the starting point. The cross sectional profile of the first line is plotted, after the grades, cubic spline values, and the segments and their end values have been calculated, as in the 'DIGCUSPDIS' program. Each of the three data types, elevation, rolling resistance and speed restriction, have their own axis that is plotted before the corresponding curve is drawn. The calculations are done for each section, the curve plotted, before the calculations are performed for the next section. Only one line is drawn per chart. The listing of the cross program appears in Appendix A1

4.1. THE USER INTERFACE

The cross-sectional program is run on the terminal connected to the Calcomp plotter by typing: "RUN [IAN.CAROL]CROSS".

As with Program II, no inputs are made while the program is being run. The input file declared at the beginning of the program must match the output file of the 'DIGCUSPDIS' program. Calcomp subroutine inputs must match what is expected. The axis length, given as a constant in the 'CALL' statement for the 'AXIS' subroutine must have the required size.

4.2. THE INPUT FILE

The input file used is the output file produced by the 'DIGCUSPDIS' program. The header data is used in generating the chart: the 'FILENAME' is used as the first part of the graph title, 'LINENAME' as the second part of the the title. The units are used as part of the 'AXES' titles. The 'LINE' data has two different sections; the 'DATA' section and the 'DISTANCE' section. The 'DATA' title is not used. The 'DATA' lines are used to produce the three different curves, one for each of the data types. The values read are used as the [y] co-ordinates. The 'DISTANCE' title forms the second part of the [x] axis title. The 'DISTANCE' data becomes the [x] co-ordinates. The 'DATA' section is ended by a line with the following format:

000 FF 0.0

The 'DISTANCE' section is ended by this flag line:

000 0.0

A detailed description of the input file format can be found in Section 3.3, the file listing is contained in Appendix A2.

4.3. THE OUTPUT FILE

At present, an output file does not exist. This program was written, however, with the purpose of providing an output file that would be used by a 'HAULAGE SIMULATION' program. At this time, that program is not completed, thus, it is not known what input data would be needed, nor what format it would assume. Until that time no output file will be produced by the 'CROSS' program.

4.4. SUBROUTINES USED

The 'CROSS' program uses both Calcomp subroutines and user-written subroutines. The Calcomp subprograms include:

PLOT: moves the plotter pen a given distance, in terms of [x] and [y]; can also define a new origin,
LINE: connects each point in an array with straight lines, and
NEWPEN: changes plotter pens.

The following user-defined subroutines are used:

CHARCOUNT: found in the 'SCALE.FOR' file. Counts the number of characters contained in a character string,
MYSCALE: found also in the 'SCALE.FOR' file; calculates a scaling factor so that all the data will fit evenly onto the axis,
CHART: plots the axes and the graph titles,
GRADE or **ANGLE:** found in the 'GRADE.FOR' file. Determines the slope at each point, and
SEGMENT: determines the number of segments that the line between two points will be divided into.

The above subroutines are described in more detail in Section 5. Listings are shown in Appendix A3.

One more subroutine is used by this program. It is the user written 'MINMAX' (x,maxx,minx) subroutine, listed at the end of the program. It is used to determine the maximum and minimum values of an array by comparing the present array element with the maximum and minimum values that are known at the time. The argument list is thus explained:

X: the array being examined,

MAXX: the maximum value of the array, and
MINX: the minimum value of the array.

The listing of this subroutine is found at the end of the program listing in Appendix A1.

4.5. GRAPHS PRODUCED

The graph produced by the 'CROSS' program is in the form of a chart; the curves are plotted within a box. All four sides have scale annotations. The graph titles are printed above the top axis. The [x] axis title is printed below the bottom axis. The [y] axis is divided into three sections:

1) the elevation curve (the cross-section profile),
2) the rolling resistance curve, and
3) the speed restriction curve.

Each section is drawn separately. The bottom axis and graph titles are drawn first, along with the elevation section. Both left and right axis are drawn, then the curve is produced. Next, the rolling resistance section is plotted, and then the speed restriction section. The top axis is drawn with the speed restriction section. Each section is a different colour. No symbols are plotted because interpolation may have occurred; the points recorded cannot be distinguished from the points that are interpolated. The [y] axis titles are plotted on alternating sides of the graph to reduce congestion of the plot. Also, the opposite side scale annotations are printed on the same side of the axis as the title.

When the title is printed on the left side of the left axis, the scale annotations are printed on the left side of the right axis as well. This is done to avoid having the first numbers of the present scale and the last numbers of the previous scale overlapping.

Finally, the speed restriction curve is drawn as a straight line between each point. This is done because the speed limit is an exact figure; it is not found by measurement and it does not change gradually. The limit is arbitrarily set and changes by an arbitrary amount.

Charts produced can be found in Appendix A4.

4.6. OTHER COMMENTS

Only one line's data is used for a chart. If more than one line exists in a file, another piece of paper must be attached to the plotter for each line that is to be plotted, and the origin of the chart moved accordingly. (The origin can be moved from within the program, in the section which performs this task, at the program's end).

At the beginning of the program, some variables have to be initialized. These include the minimum values of the three [y] axis arrays (elevation, rolling resistance and speed restrictions). They are set equal to <1,000,000> and the maximums of the same arrays are set at <-1,000,000>. If the input data does not fall within this range, these values must be changed.

5. SUBROUTINES: USER-WRITTEN SUBPROGRAMS

Subroutines are small programs which perform a specific duty, be it a certain set of calculations or a number of manipulations. Usually, they are called on more than once in a single program, or by several programs simultaneously. Two types of subroutines are used by the programs described in this report; user-written and Calcomp software. The latter are presented in Appendix A5, the listings in Appendix A3.

Fourteen user-written subroutines are employed by the two '**CUBIC SPLINE**' programs. Another is used by the '**CROSS**' program, '**MINMAX**', but it pertains only to that program (its listing is found at the end of the program); it will not be described here. The subroutines used by the '**DIGIT**' program have also already been described.

5.1. THE CHARCOUNT SUBPROGRAM: '**CHARCOUNT**(string,n)'

This subroutine determines the length of a character string. String elements are compared until a double blank is encountered. A double blank is considered the end of the string. At that point, the subroutine terminates and returns an integer value to the calling program. Only two arguments are required by '**CHARCOUNT**':

STRING:	a character string, and
N:	the character string length (number of characters in the string)

The listing of '**CHARCOUNT**' appears in Appendix A3.

5.2. THE MYSCALE SUBROUTINE: '**MYSCALE** (array,n,min,max,length,first,fact)'

The '**MASCALE**' subroutine scales an array of data to fit evenly on a given axis length. It has seven arguments:

ARRAY:	an array of data (real numbers) that is to be scaled,
N:	the number of elements in the array,
MIN:	the minimum value of the array,
MAX:	the maximum value of the array,
LENGTH:	the axis length,
FIRST:	first value that appears on the axis scale, and
FACT:	the scaling factor.

Only the last two arguments are changed before being returned to the main program. The first value is found by rounding the minimum value down to the nearest whole number. The scaling factor is found by subtracting the first value from the largest value (the maximum value rounded up to the nearest whole number), then dividing by the length:

e.g. FACT = [(LARGEST - FIRST)] / LENGTH ..... (3)

The subprogram listing is also found in Appendix A3.

5.3. THE DATAPLT SUBROUTINE: 'DATAPLT (x,y,n,sym)'

This subroutine plots only the symbols of the points formed by the x and [y] arrays, by calling the line subroutine. If more than one line is plotted per graph, the symbol of each line is changed. Four arguments are needed:

X,Y: data points to be plotted,
N: number of data points to be plotted, and
SYM: integer value of the symbol to be plotted.

The listing of '**DATAPLT**' is found in Appendix A3.

5.4. THE GRADE SUBROUTINE; 'GRADE (x,y,n,j,s)'

'**GRADE**' is called upon to calculate the slope of each point of the [x] and [y] arrays. It has the following argument list:

X: [x] co-ordinate array,
Y: [y] co-ordinate array,
N: number of points in each array,
J: the point whose slope is to be calculated, and

S: slope that is calculated.

The slopes for the first and last points are calculated using the formula:

$$S = (Y_2 - Y_1) / (X_2 - X_1) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (4)$$

The slope for all the points in between is calculated by using the points adjacent to the point being considered, such as:

$$S = [Y\ (after) - Y\ (before)]\ /$$
$$[X\ (after) - X\ (before)[\quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (5)$$

If <X(after) = X(before)>, then the slope is assigned a value of <1,000,000>, which is very close to the vertical slope required. The listing of grade is located in Appendix A3.

5.5. THE ANGLE SUBROUTINE: 'ANGLE (x,y,n,j,s)'

The 'ANGLE' subroutine was written as an alternative method of calculating the slope at each point, using angles. The argument list is the same as the one required by the 'GRADE' subroutine. Unfortunately, in its present form is not used. It is included because it offers a potentially elegant solution.

First, the slope for the first and last points are calculated in exactly the same way as they were in the grade subroutine.

$$S = (Y_2 - Y_1) / (X_2 - X_1) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (6)$$

If $< X_2 = X_1 >$, the slope was assigned a value of <1,000,000>. Next, the distances between the first two points, [d1], and the second and third points, [d2], are calculated. The angles formed by the hypotenuses [d1] and [d2] and the [y] axis are calculated, [tau] and [mu] respectively. If [d2=d1], [tau=mu] and [X1=X2], then the slope is assigned a value of <1,000,000> (a vertical slope).

For the same conditions, except that [Y1=Y2]instead of [X1=X2], the slope obtained a value of <0.0> (a horizontal slope). It was now found that eight different cases must be considered before the slope could be calculated:

1) Y1 > Y2 < Y3,
2) Y1 < Y2 > Y3,
3) (X1 > X2 > X3) and (Y1 > Y2 > Y3),
4) (X1 > X2 > X3) and (Y1 < Y2 < Y3),
5) X1 < X2 > X3,
6) X1 > X2 < X3,
7) (X1 < X2 < X3) and (Y1 < Y2 < Y3), and
8) (X1 < X2 < X3) and (Y1 > Y2 > Y3).

Each case was further divided into four conditions:

1) TAU < (PI / 8),..... (7)
2) (PI / 8) < TAU < (PI / 4),..... (8)
3) (PI / 4) < TAU < [(3 * PI) / 8], and ..... (9)
4) TAU < [(3 * PI) / 8]..... (10)

Figure 2 illustrates the basic equation from which all the succeeding calculations were derived:

$$[A/B] = [D_1 / D_2] \quad \text{.....} \quad (11)$$

Mathematical manipulations, based on the drawings in Figure 3, resulted in the following equations, calculating the angle corresponding to the slope required (ALPHA):

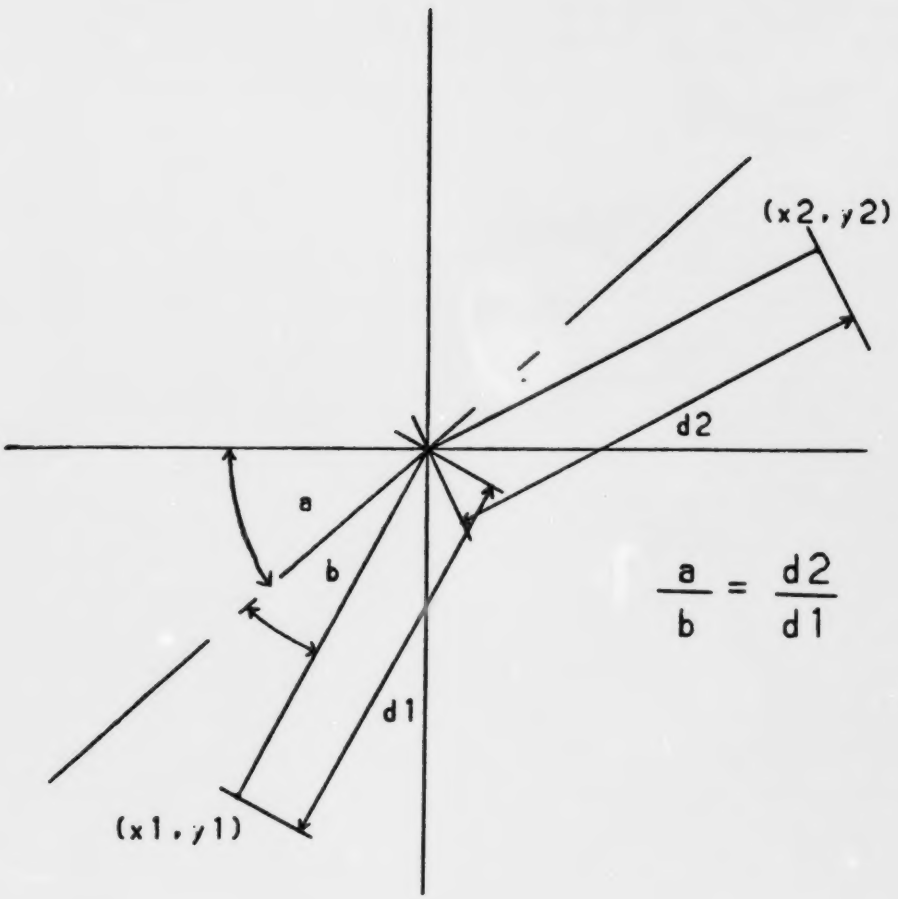


Fig. 2 - Angle Subroutine's Basis

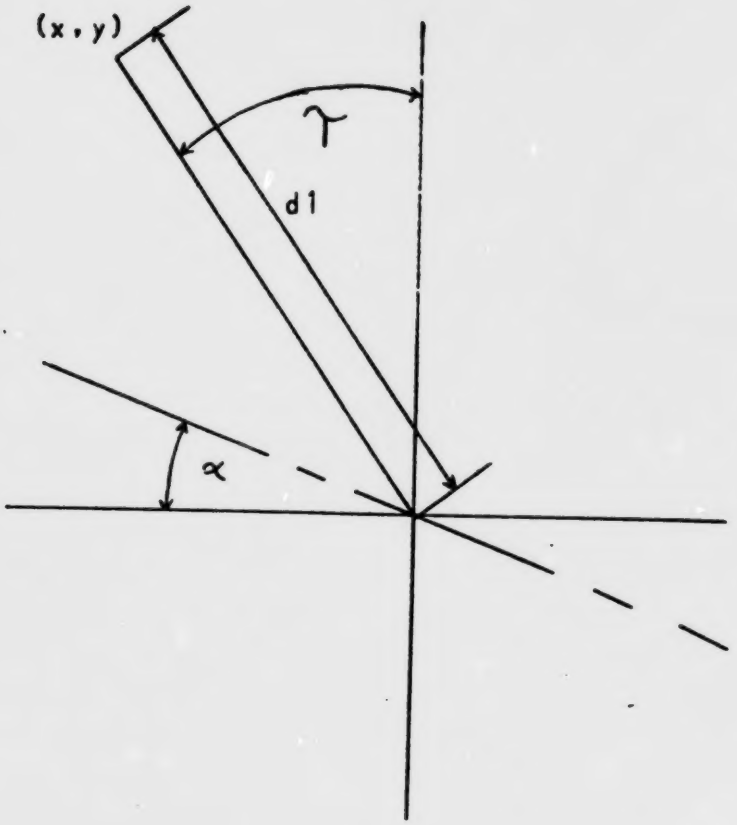


Fig. 3 - Calculating Alpha: Case 1

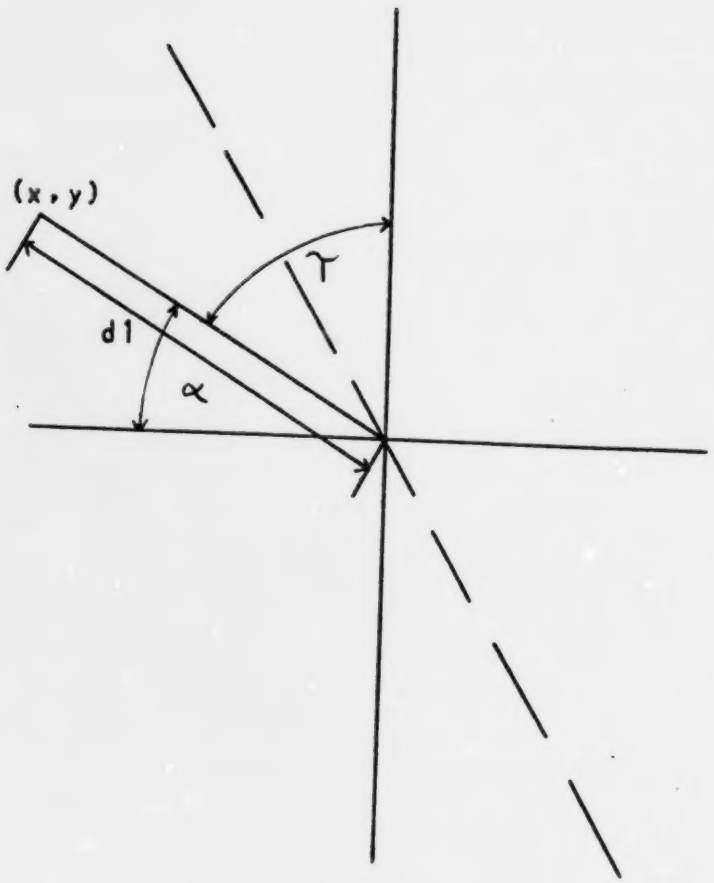


Fig. 4 - Calculating Alpha: Case 2

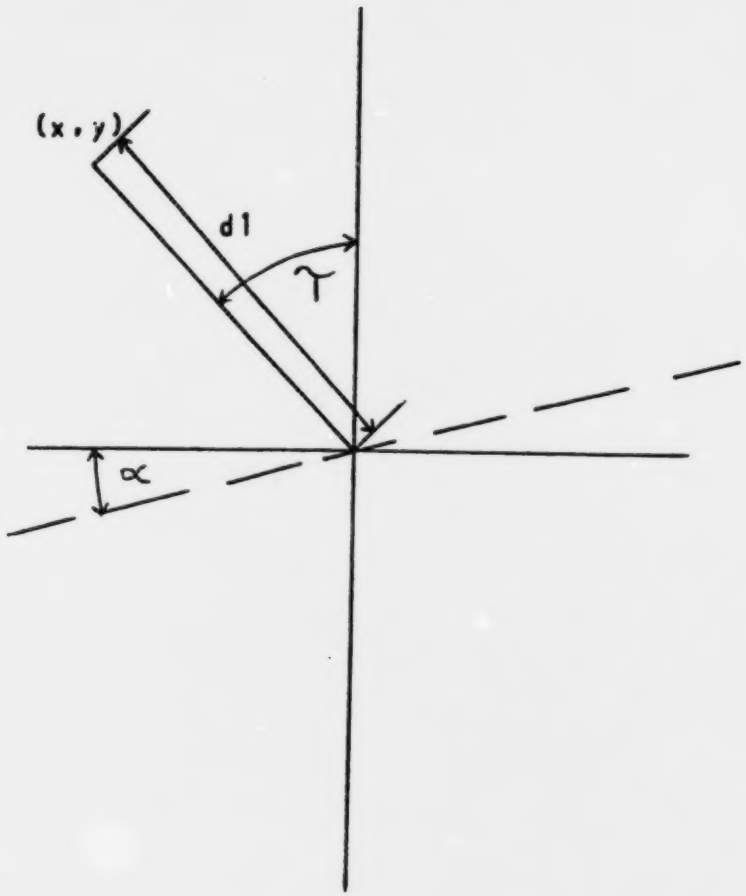


Fig. 5 - Calculating Alpha: Case 3

1. ALPHA = [(PI / 2 - TAU) * D_2] / [D_2 + D_1] ..... (12)
2. ALPHA = [(PI / 2 - TAU) * D_2] / [D_2 - D_1] ..... (13)
3. ALPHA = [(TAU - PI /2) * D_2] / [D_2 - D_1] ..... (14)

At this time, it is not known exactly which equation fits where in each case and condition. As well, with the subroutine not working, it is not known if there is an error in the basic logic, an error in the calculations or if the division into cases and conditions is not taken far enough.

A graph produced by the angle subroutine is contained in Appendix A4. The subroutine listing is in Appendix A3.

5.6. THE TRANSFORM SUBROUTINE:

'TRANSFORM'(p,x,y,j,e,q,r,n,theta,tx,ty,nq,nr)

The **'TRANSFORM'** subroutine has two purposes. Firstly to calculate the angle of the axis rotation when a data point set is transformed to a new co-ordinate system, secondly is to transform each individual point of a set to a new system. The argument list includes:

P: integer that determines if **'THETA'** must be calculated,
X,Y: real arrays that contain the data points,
J: index number of the **'DO'** loop, from which transform is called,
E: number of points to be transformed,
Q,R: [x] and [y] offsets (distance from the origin that the new origin will be offset),
THETA: the angle of axis rotation,
Tx,Ty: the transformed data points, and
Nq,Nr: the transformed [x] and [y] offsets

The axis rotation is done by calling the subroutine **'THETAA'**.

'**THETAA** (x,q,y,r,theta)'

'**THETA**' is calculated by the following equation:

THETA = ARCTAN [(Y - R) / (X - Q)] ..... (15)

The [x] and [y] offsets are now calculated in terms of the new system. The procedure used is the same as that used for point transformation which is described next.

The point transformation is more complicated than the axis rotation. The hypotenuse between each point is calculated, then the angle between the hypotenuse and the [x] axis is found by calling '**BETAA**'.

'**BETAA** (x,q,y,r,beta)'

'**BETAA**' finds the angle using the same equation that thetaa does, but betaa is used instead, because theta is returned to the main program whereas beta is not. Once beta has been calculated, gammaa is called.

GAMMAA(x,q,beta,theta,gamma)

'**GAMMAA**' determines the angle between the hypotenuse and the new [x] axis.

GAMMA = BETA - THETA ..... (16)

The points are then transformed by multiplying the hypotenuse by the cosine of gamma for the [x] co-ordinate; by the sine of gamma for the [y] co-ordinate.

The listing of the '**TRANSFORM**', '**THETAA**', '**BETAA**', and '**GAMMAA**' sub-routines are found in Appendix A3.

5.7. THE ANGSLOPE SUBROUTINE: 'ANGSLOPE (theta,s1,s2,ns1,ns2)'

The 'ANGSLOPE' subroutine converts a slope to an angle, the angle to a new co-ordinate system, then changes the new angle back to a slope. The arguments are as follows:

THETA: the angle of axis rotation (found in the 'TRANSFORM' subroutine),
S1,S2: the original slopes of points 1 and 2, and
NS1,NS2: the slopes of points 1 and 2 in a new co-ordinate system.

The angle of the slope is found by taking the arctangent of the slope. The new angle equals the old angle minus theta. The new slope is the tangent of the new angle.

ANGLE = ARCTAN (SLOPE) ..... (17)
NEW ANGLE = ANGLE - THETA ..... (18)
NEW SLOPE = TAN (NEW ANGLE) ..... (19)

The listing of 'ANGSLOPE' is in Appendix A3.

5.8. THE SPLINE SUBROUTINE: 'SPLINE (x1,x2,y1,y2,s1,s2,a,b,c,d)'

X1,Y1,X2,Y2: the first and second data points,
S1,S2: slopes of the first and second data points, and
A,B,C,D: constants of the cubic spline equation.

The 'SPLINE' subroutine calculates the constants of the cubic spline equation. The following equations were used:

$$Y_1 = A + (B * X_1) + (C * X_1^2) + (D * X_1^3) \quad \ldots\ldots\ldots\ldots\ldots \quad (20)$$

$$Y_2 = A + (B * X_2) + (C * X_2^2) + (D * X_2^3) \quad \ldots\ldots\ldots\ldots\ldots \quad (21)$$

$$S_1 = B + (2 * C * X_1) + (3 * D * X_1^2) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (22)$$

$$S_2 = B + (2 * C * X_2) + (3 * D * X_2^2) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (23)$$

Mathematical manipulation resulted in the equations that are used to solve all four constants. The equations are complicated, they can be found in the subroutine's listing in Appendix A3.

5.9. THE SEGMENT SUBROUTINE: 'SEGMENT (x,y,j,n,k)'

The 'SEGMENT' subroutine divides the line between two points into segments. The number of segments is determined by the distance between the [x] co-ordinates or the [y] co-ordinates:

- **X,Y:** the [x] and [y] co-ordinates,
- **J:** the index of the do loop from which segment is called,
- **N:** the number of data points in the line, and
- **K:** the number of segments the line between two points is divided into.

The number returned to the main program is arbitrarily set in the 'SEGMENT' subroutine. The subroutine's listing is found in Appendix A3.

5.10. THE CUGRAPH SUBROUTINE:

'CUGRAPH (firstx,factx,firsty,facty,namea,nameb,xaxis,yaxis)'

The 'CUGRAPH' subroutine plots the graph titles and the axes for a graph:

FIRSTX,FIRSTY:	the first values of the [x] and [y] axes,
FACTX,FACTY:	the scaling factors used in plotting the [x] and [y] axes,
NAMEA,NAMEB:	the names used in the two graph title lines, and
XAXIS,YAXIS:	the names used for the [x] and [y] axis titles.

'CUGRAPH' first determines the number of characters in each title, for all of the titles, by calling 'CHARCOUNT'. Then it calls 'SYMBOL' to plot the graph titles. Last, it calls 'AXIS' to plot the axes, the axes' titles and the scale annotations. The 'CUGRAPH' subroutine is listed in Appendix A3.

5.11. THE CHART SUBROUTINE: 'CHART (pen,x,y,top,r,n,namea,nameb, yaxis,xaxis,topaxis,raxis)'

PEN:	the number of the pen to be used in plotting,
X,Y,TOP,R:	the arrays of co-ordinates for the bottom, left, top, and right axes, respectively,
NAMEA,NAMEB:	the names used in the graph title lines, and
XAXIS,YAXIS,TOPAXIS,RAXIS:	the names used in the axes' titles.

The 'CHART' subroutine plots a chart (four-sided graph). It can be modified, but for now the chart is made up of three sections. The first time 'CHART' is called, (it must be called once for each section), it plots the bottom or [x] axis and the first section. The plotter is initialized by a 'CALL TO PLOT', an origin is established by a 'CALL TO PLOT', and the title is drawn by a 'CALL TO SYMBOL'. The axes, both right and left, for section one are then drawn. The second call for chart produces only the second section axes. The last call to chart produces the third section axes and the

top axis. With each call that is made for chart, the character string length for the titles passed is calculated.

The listing for 'CHART' appears in Appendix A3.

6. CONCLUSION

In this report, a set of programs used to digitize and plot mine roads, is described. The method employed in the running of these programs is detailed, as are calculations and subroutines that are used. The structure of input/output files, graphs, and charts produced are also described.

7. RECOMMENDATIONS

Once a 'HAULAGE SIMULATION' program is created, the 'CROSS' program will have to create an output file. This should include header data such as the following:

-filename,
-[x] value range,
-[y] value range,
-north direction arrow co-ordinates, and
-units of every data type (array).

The line data should include the linename(s), elevation, rolling resistance, speed restriction and distance arrays.

A second output file from the 'DIGCUSPDIS' program may also be required. This file would consist of the header data given above and the points digitized for each line.

The last recommendation is that some modification should be done to the north direction arrow, purely for aesthetic reasons. This could be done by adding more points to the arrow, plotting the first part without symbols, the second part with symbols. Possibly, the symbol used could be changed as well.

8. APPENDICES

(A1 - A4)

A1 - PROGRAM LISTINGS

		Page
1.	Digit	34 - 37
2.	Digcuspdis	38 - 42
3.	Cross	43 - 48

```
      program digit

c     program that reads data from the digitizer into a data file

      real x,y,xo,yo,startx,starty,factx,facty,endx,endy,zdata
      integer n
      character*30 filename,button,but,linename,units,elunits,ty,addat
      character*30 srunits,rrunits

c     x,y=data points
c     n=# of points digitized, including the origin
c     filename=name of data file
c     button=digitizer button pressed, which determines where the point
c            is in the line
c     but=first and only needed element of button
c     xo,yo=co-ordinates of the origin
c     linename=name of line
c     zdata=additional data that is required
c     endx,endy=last value for scale
c     startx,starty=first value for scale
c     units=units used for data
c     elunits=elevation units used
c     srunits=speed restriction units
c     rrunits=rolling resistance units
c     factx,facty=scale factor
c     ty=type of additional data entered
c     addat=determines where to move within program after additional da
c              has been entered

c     name data file

      type*,'enter data file name, press return <CR>'
      read(5,10),filename
10    format(a)

c     open data file

      open(unit=1,file=filename,status='unknown')

c     write filename to file

      write(1,*),'filename:  ',filename

c     read in data

c     define origin

      n=1
      factx=1
      facty=1
      type*,'first point digitized will be origin, use 0 button'
      read(5,20,end=1000),button,x,y
      call calc(n,x,y,xo,yo,button,but,factx,facty,startx,starty)

c     determine x-maximum

      type*,'next point is the x-maximum, use button 9'
      read(5,20),button,x,y
      call calc(n,x,y,xo,yo,button,but,factx,facty,startx,starty)

c     determine x scale values

      type*,'type in start and end of x-scale, "XXX.XXX" <CR>'
      read(5,100),startx,endx
100   format(2(f7.3,1x))
90    format(2(1x,f7.3))
```

```
      write(1,*)'x-value range:  ',startx,' - ',endx
      call scale(x,startx,endx,factx)

c     determine y maximum

      type*,'next point is the y-maximum, use button 8'
      read(5,20),button,x,y
      call calc(n,x,y,xo,yo,button,but,factx,facty,startx,starty)

c     determine y scale values

      type*,'type in start and end of y-scale, "YYY.YYY"'
      read(5,100),starty,endy
      write(1,*),'y-value range:  ',starty,' - ',endy
      call scale(y,starty,endy,facty)

c     determine which direction is north

      type*,'next point indicates direction of data.  Digitize 2 points i
     3 top right-hand corner to form a north arrow, use button 1'
      read(5,20),button,x,y
      call calc(n,x,y,xo,yo,button,but,factx,facty,startx,starty)
      write(1,*)'north arrow, point 1:  ',x,y
      read(5,20),button,x,y
      call calc(n,x,y,xo,yo,button,but,factx,facty,startx,starty)
      write(1,*)'north arrow, point 2:  ',x,y

c     define variable units

      type*,'type in name of units'
      read(5,10),units
      write(1,*),'plot units:  ',units

      type*,'type in name of elevation units'
      read(5,10),elunits
      write(1,*),'elevation units:  ',elunits

      type*,'type in name of rolling resistance units'
      read(5,10),rrunits
      write(1,*),'rolling resistance units:  ',rrunits

      type*,'type in speed restriction units'
      read(5,10),srunits
      write(1,*),'speed restriction units:  ',srunits

c     digitize each line data

77    type*,'to start a line use A button, to continue a line without add
     3 data use B button, to end a line use D button, to end the file us
     3 the # button, to continue a line with data use any other button'

20    format(a2,f6.2,f6.2)

      read(5,20),button,x,y
      call calc(n,x,y,xo,yo,button,but,factx,facty,startx,starty)

c     name each line

      if(but.eq.'A')then
          type*,'enter line name'
          read(5,10),linename
          write(1,*)
          write(1,40),linename
40        format(1x,a)
      endif
```

```
c       write data to file

        write(1,50),but,x,y
50      format(1x,a1,f7.2,f7.2)

        if(but.eq.'B')then
            goto 77
        endif

c       write the zdata to the output file

        type*,'enter additional data:'
        type*,'a for rolling resistance'
        type*,'b for elevation'
        type*,'c for speed restrictions'
        type*,'d to go to next point'

        read(5,30),addat
30      format(a)

47      if(addat.eq.'a')then
            type*,'type RR and data for rolling resistance'
            type*,'if more data to follow, type & at beginning of line;
        3 type % at beginning if no more to follow'
            read(5,70),sign,ty,zdata
70          format(a1,1x,a2,1x,f10.3)
            write(1,60),sign,ty,zdata
60          format(2x,a1,2x,a2,1x,f10.3)
            type*,'type b for elevation, c for speed restrictions or d to
        3 go to next point'
            read(5,30),addat
            goto 47
        elseif(addat.eq.'b')then
            type*,'type & at beginning if more to follow, % if done'
            type*,'type EL and data for elevation'
            read(5,70),sign,ty,zdata
            write(1,60),sign,ty,zdata
            type*,'type c for speed restrictions or d to go to next point'
            read(5,30),addat
            goto 47
        elseif(addat.eq.'c')then
            type*,'type %, SR and data for speed restriction'
            read(5,70),sign,ty,zdata
            write(1,60),sign,ty,zdata
            type*,'type d to go to next point, type e to end line or type f
        3 end file'
            read(5,30),addat
            goto 47
        elseif(addat.eq.'d')then
            goto 77
        elseif(addat.eq.'e')then
            goto 77
        elseif(addat.eq.'f')then
           type*,'last point recorded in file'
            goto 1000
        endif

c       close data file

1000    close(unit=1)

        stop
        end

c*******************************************************************************
```

```
      subroutine calc(n,x,y,xo,yo,button,but,factx,facty,startx,starty)

c     a subroutine to calculate data points with regard to the origin
c     and to shorten the variable button to but for writing to file

      real x,y,xo,yo,factx,facty,startx,starty
      integer n
      character*30 button,but

c     n=number of times calc subroutine has been called
c     x,y=data points
c     xo,yo=co-ordinates of uncrrected origin
c     button=characters used to distinguish what each point represents
c     but=first character of button
c     factx,facty=correct value of each point
c     startx,starty=initial values of x and y

c     define origin

      if(n.eq.1)then
          xo=x
          yo=y
      endif

      n=n+1

c     subtract origin from data points

      x=x-xo
      y=y-yo

c     find true value of points

      if(n.gt.3)then
          x=(x*factx)+startx
          y=(y*facty)+starty
      endif

c     truncate button

      but=button(1:1)

c     return to main program

      return
      end

c*************************************************************************************

      subroutine scale(x,start,end,fact)

c     subroutine to calculate scaling factors

      real x,start,end,fact

c     x=data point
c     start=first point along scale
c     end=last point along scale
c     fact=scale factor

      fact=(end-start)/x

c     return to main program

      return
      end
```

```
      program digcuspdis

      real x(1000),y(1000),a(1000),b(1000),c(1000),d(1000),s(1001),ex(1000
      real why(1000),tx(1000),ty(1000),theta,ns(1000),q,r,nx(1000),ny(1000
      real nq,nr,minx,miny,maxx,maxy,dat,hyp(1000),distance,factx
      real facty,firstx,firsty,directx(4),directy(4)
      integer n,k,j,p,e,sym,unit,lin
      character*75 namea,nameb,xaxis,yaxis,but,linename,units,loop,type

c     program to plot curves by fitting a cubic curve to given data
c     and calculate the distance between two points

c     x,y=original co-ordinates
c     n=number of lines in each line
c     a,b,c,d=parameters for cubic spline equation
c     s=slopes at points
c     ex,why=points calculated for cubic equation for each segment
c               between every two points
c     tx,ty=transformed points used to calcuate ex and why
c     theta=axis rotation (in rads) used to transform points
c     ns=new slope provided when transformed to new axis
c     q=x-offset used in transformation
c     nq=x-offset in terms of new origin
c     r=y-offset used in transformation
c     nr=y-offset in terms of new origin
c     nx,ny=new co-ordinates of ex and why when transformed back to origin
c               axis
c     k=# of segments a line between two points is divided into
c     namea=first line of graph title
c     nameb=second line of graph title
c     xaxis=x-axis title
c     yaxis=y-axis title
c     p=constant to determine if theta has to be calculated or is given
c     e=# of points to be transformed
c     sym=# of symbol that is to be plotted
c     but=digitizer button that determines where the point is on a line
c     minx,miny=minimum values of x and y, used in calculation
c             of scale factors
c     maxx,maxy=maximum values of x and y, used in calculation of
c             scale factors
c     linename=name of each line
c     dat=additional data (e.g. elevation)
c     type=type of additional data present
c     hyp=distance between eachpoint calculated between two given points
c     distance=distance along curve between two given points
c     directx,directy=data points for direction arrow
c     factx,facty=scaling factors
c     firstx,firsty=first values on each scale
c     unit=units loop count
c     units=units of additional data
c     loop=line character to determine if more data follows
c     lin=# of characters in line name

c     open an output file, input file and an output file for the
c     additional data

      open(unit=3,file='zdata.dat',status='new')
      open(unit=2,file='out.dat',status='new')
      open(unit=1,file='input.dat',status='unknown')

c     initialize variables as needed, read in data from input file

      sym=0

c     read filename, assign it to graph title and send to output file
```

```
        read(1,20),namea
20      format(12x,a)
        call charcount(namea,lin)
        write(3,120),namea(1:lin)
120     format(' filename:  ',a)

c       read in x maximum and minimum

        read(1,60),minx,maxx
60      format(17x,e15.7,3x,e15.7)

c       read in y maximum and minimum

        read(1,60),miny,maxy

c       read in data direction

        read(1,130),directx(1),directy(1)
        read(1,130),directx(2),directy(2)
130     format(24x,2f11.7)
30      format(a)

c       read in plot units and assign to x and y axis' names
c       send to output file as well

        read(1,50),xaxis
50      format(14x,a)
        call charcount(xaxis,lin)
        write(3,133),xaxis(1:lin)
133     format(' distance units:  ',a)
        yaxis=xaxis

c       read in data units and assign to appropiate output file

        unit=1
53      read(1,30),units
        write(3,30),units
        unit=unit+1
        if(unit.lt.4)then
            goto 53
        endif

c       read in line name, send to output file, and assign to graph title

73      read(1,30)
        read(1,30),linename
        write(3,30),linename
        write(3,*),'DATA'
        call charcount(linename,lin)
        if(sym.ne.0)then
            nameb=linename(1:lin)//', '//nameb
        else
            nameb=linename(1:lin)
        endif

c       initialize n

        n=1

c       read in point data

100     read(1,10,end=1000),but,x(n),y(n)
10      format(1x,a1,2f7.2)

c       interpret digitizer buttons
```

```
c       read in additional data when a new line is started

        if(but.eq.'A')then
            sym=sym+1
23          read(1,70),loop,type,dat
70          format(2x,a1,2x,a2,1x,f10.3)
            write(3,80),n,type,dat
80          format(1x,i3,1x,a2,1x,f10.3)
            if(loop.eq.'&')then
                goto 23
            endif

c       if end-of-file label is reached

        elseif((but.eq.'D').or.(but.eq.'#'))then
13          read(1,70),loop,type,dat
            write(3,80),n,type,dat
            if(loop.eq.'&')then
                goto 13
            endif

c       write an end flag to output file

            write(3,30),' 000 FF 0.0'
            write(3,30),' DISTANCE'
            goto 1000

c       if some additional data is present

        elseif((but.ne.'A').and.(but.ne.'D').and.(but.ne.'#').and.
        3 (but.ne.'B'))then
43          read(1,70),loop,type,dat
            write(3,80),n,type,dat
            if(loop.eq.'&')then
                goto 43
            endif
        endif

        n=n+1
        goto 100
1000    continue

c       plot symbols for all lines

        if(sym.eq.1)then
            call plots(0,0,6)
            call plot(0.0,0.0,-3)
            call myscale(x,n,minx,maxx,15.5,firstx,factx)
            call myscale(y,n,miny,maxy,21.0,firsty,facty)
            x(n+1)=firstx
            x(n+2)=factx
            y(n+1)=firsty
            y(n+2)=facty
            call dataplt(x,y,n,sym)

c       plot direction arrow

            directx(3)=firstx
            directx(4)=factx
            directy(3)=firsty
            directy(4)=facty
            call line(directx,directy,2,1,1,2)
            call symbol(15.,19.6,0.21,'N',14,0.,0)
        else
            x(n+1)=firstx
            x(n+2)=factx
```

```
            y(n+1)=firsty
            y(n+2)=facty
            call newpen(sym)
            call dataplt(x,y,n,sym)
        endif

c       move pen back to origin

        call plot(0.,0.,3)

c       perform cubic function and plot curve

c       calculate values to pass to the spline subroutine

        do 5, j=1,n-1
            call grade(x,y,n,j,s)
5       continue

        do 25, j=1,n-1

c       initialize transform subroutine parameters

            q=x(j)
            r=y(j)
            e=0
            p=0

c       transform two points to new origin

            call transform(p,x,y,j,e,q,r,n,theta,tx,ty,nq,nr)

c       transform slopes to new co-ordinates

            call angslope(theta,s(j),s(j+1),ns(j),ns(j+1))

c       calculate cubic equation constants

            call spline(tx(1),tx(2),ty(1),ty(2),ns(j),ns(j+1),a(j),
     3          b(j),c(j),d(j))

c       determine # of segments that each line will be split into

            call segment(tx,ty,j,n,k)

c       transfer scale parameters to cubic function calculations
c       so that the curves can be plotted (using new co-ordinates
c       after transforming back to original co-ordinates)

            nx(k+2)=x(n+1)
            nx(k+3)=x(n+2)
            ny(k+2)=y(n+1)
            ny(k+3)=y(n+2)

c       calculate the values of points along the line segments

            ex(1)=tx(1)
            why(1)=ty(1)

            do 35, i=1,k
                ex(i+1)=tx(1)+(i*(tx(e+2)-tx(1)))/(k)
                why(i+1)=(a(j))+(b(j)*ex(i+1))+(c(j)*(ex(i+1)**2))+(d(j)*
     3                  (ex(i+1)**3))
35          continue

c       calculate the distance between two points (following the curve)
```

```
            distance=0.0
            do 75,i=1,k
                hyp(i)=sqrt(((ex(i+1)-ex(i))**2)+((why(i+1)-why(i))**2))
                distance=distance+hyp(i)
75          continue

c       write distance to file

            write(3,150),j,distance
150         format(1x,i3,1x,f7.2)

c       transform points back to original co-ordinates,after changing parame

            e=k+1
            p=1
            call transform(p,ex,why,j,e,nq,nr,n,-theta,nx,ny,q,r)

c       plot curve

            call line(nx,ny,k+1,1,0,4)

25      continue

c       if digitizer is used for more than one line go back to beginning
c       for next line

        if(but.eq.'D')then

c       send a line end label to output file

            write(3,30),' 000 0.0'
            goto 73
        endif

c       plot graph titles and axes

        call newpen(1)
        call cugraph(firstx,factx,firsty,facty,namea,nameb,xaxis,yaxis)

c       end of plot

        call plot(16.0,0.0,999)

        stop
        end
```

```
      program cross

      real x(1000),y(1000),a(1000),b(1000),c(1000),d(1000),s(1001),ex(1000)
      real why(1000),firstx,firstel,firstrr,firstsr,factx,factel,factrr
      real factsr,minx,minel,minrr,minsr,maxx,maxel,maxrr,maxsr,data
      real distance(1000),EL(1000),RR(1000),dist,SR(1000)
      integer interel,interrr,intersr,indexel,indexrr,indexsr,n,k,eln
      integer rrn,srn,pen,point,pointd,count
      character*75 namea,nameb,xaxis,xaxis1,xaxis2,yaxis1,yaxis2,yaxis3
      character*75 taxis,raxis,type

c     program to plot curves by fitting a cubic curve to given data
c     does not fit curves to data which x2<x1 (no overhang or circles)

c     x,y-data points
c     EL,RR,SR-data for elevation, rolling resistance and speed restriction
c     a,b,c,d-spline constants
c     ex,why-calculated points on each segment between two points
c     n-# of data lines in file
c     eln,rrn,srn-# of lines in each array that have been read from input
c     point,pointd-point at which data for EL,RR,SR,distance arrays
c           has been read
c     type-which array data read is placed into
c     data-data read to be placed in an array (EL,RR,SR)
c     k-# of segments each line between two points is divided into
c     namea,nameb-title of graph
c     xaxis,xaxis1,xaxis2,yaxis1,yaxis2,yaxis3,taxis,raxis-names of axes
c     firstx,firstel,firstrr,firstst-first value for x,EL,RR and SR axis
c     factx,factel,factrr,factsr-scaling factors for x,EL,RR and SR axis
c     minx,minel,minrr,minst-minimum values of x,EL,RR,SR
c     maxx,maxel,maxrr,maxsr-maximum values of x,EL,RR,SR
c     pen-the integer that indicates which graph is being plotted, eg rr
c     distance-distance from starting point
c     dist-distance between points, read from input file
c     indexel,indexrr,indexsr-last point in array which was read,
c         not interpolated
c     interel,interrr,intersr-point which is to be interpolated
c     count-# of characters in xaxis2

c     read in data from file and determine # of lines in file and
c     open a output file if necessary

      open(unit=2,file='output.dat',status='new')
      open(unit=1,file='zdata.dat',status='unknown')

c     read in filename and assign to graph title

      read(1,10),namea
10    format(12x,a)

c     read in distance units, assign to xaxis

      read(1,20),xaxis1
20    format(18x,a)

c     read in elevation units, assign to yaxis1

      read(1,30),yaxis1
30    format(19x,a)
      yaxis1='elevation in '//yaxis1

c     read in rolling resistance units, assign to yaxis2

      read(1,40),yaxis2
40    format(28x,a)
      yaxis2='rr in '//yaxis2
```

```
c       read in speed restriction units, assign to yaxis3

        read(1,50),yaxis3
50      format(27x,a)
        yaxis3='speed rest. in '//yaxis3

c       read in linename, assign to nameb

333     read(1,60),nameb
60      format(a)

c       initialize variables

        n=2
        eln=0
        rrn=0
        srn=0
        minx=0.
        minel=1000000.
        minrr=1000000.
        minsr=0.
        maxel=-1000000.
        maxrr=-1000000.
        maxsr=0.0
        distance(1)=0.
        indexel=0.
        indexrr=0.
        indexsr=0.
        pen=1
        taxis='_ '
        raxis='_ '

c       read in data title, it is not used

        read(1,60)

c       read in data

100     read(1,70,end=1000),point,type,data
70      format(1x,i3,1x,a2,1x,f10.3)

c       assign data to proper arrays and determine minimum and maximum value
c       interpolate the values between the given values where necessary

        if(type.eq.'EL')then

c       assign to elevation array

            EL(point)=data

c       determine maximum and minimum values

            call minmax(EL(point),maxel,minel)

c       interpolate data, if needed

            interel=1
33          if((eln.ne.0).and.(point.ne.(interel+indexel)))then
                EL(indexel+interel)=(((EL(point)-EL(indexel))*interel)/
3                   (point-indexel))+EL(indexel)
                interel=interel+1
                goto 33
            endif
            indexel=point
            eln=eln+1
```

```
      elseif(type.eq.'RR')then
          RR(point)=data
          call minmax(RR(point),maxrr,minrr)
          interrr=1
43        if((rrn.ne.0).and.(point.ne.interrr+indexrr))then
              RR(indexrr+interrr)=(((RR(point)-RR(indexrr))*interrr)/
     3            (point-indexrr))+RR(indexrr)
              interrr=interrr+1
              goto 43
          endif
          indexrr=point
          rrn=rrn+1
      elseif(type.eq.'SR')then
          SR(point)=data
          call minmax(SR(point),maxsr,minsr)
          intersr=1
53        if((srn.ne.0).and.(point.ne.intersr+indexsr))then
              SR(indexsr+intersr)=SR(indexsr+intersr-1)
              intersr=intersr+1
              goto 53
          endif
          indexsr=point
          srn=srn+1
      elseif(type.eq.'FF')then

c     this is the flag that tells that all the additional data has been re

          goto 1000
      endif

      goto 100
1000  continue

c     read in distance title, assign to xaxis

      read(1,60),xaxis2
      call charcount(xaxis2,count)
      xaxis=xaxis2(1:count)//'in '//xaxis1

c     read in distance data, find distance for each point and find
c     distance maximum

200   read(1,80,end=2000),pointd,dist
80    format(1x,i3,1x,f7.2)

      if(pointd.eq.000)then

c     this is the flag that shows that all the distance data has been read

         goto 2000
      endif

      distance(n)=distance(n-1)+dist
      n=n+1
      goto 200
2000  continue

      n=n-1
      maxx=distance(n)

c     Plot x-axis and titles of graph, calculate xaxis scale.

      call myscale(x,n,minx,maxx,15.5,firstx,factx)
      x(n+1)=firstx
      x(n+2)=factx
      do 35,i=1,n
```

```
            x(i)=distance(i)
35      continue

c       Plot elevation axis and  calculate scale.

        call myscale(EL,n,minel,maxel,6.5,firstel,factel)
        EL(n+1)=firstel
        EL(n+2)=factel
        call chart(pen,x,EL,x,EL,n,namea,nameb,xaxis,yaxisl,taxis,raxis)
        do 45, i=1,n+2
            y(i)=EL(i)
45      continue

c       perform cubic function and plot curve
c       calculate values to pass to the spline subroutine

63      do 5, i=1,n
            call grade(x,y,n,i,s)
5       continue

        do 25, j=1,n-1

c       calculate cubic spline coefficients

            call spline(x(j),x(j+1),y(j),y(j+1),s(j),s(j+1),a(j),
3           b(j),c(j),d(j))

c       determine the number of segments that each line will be split into

            call segment(x,y,j,n,k)

c       calculate the values of points along the line segments

            ex(1)=x(j)
            why(1)=y(j)
            do 15, i=1,k
                ex(i+1)=x(j)+(i*(x(j+1)-x(j)))/(k)
                why(i+1)=(a(j))+(b(j)*ex(i))+(c(j)*(ex(i)**2))+(d(j)*
3                   (ex(i)**3))
15          continue

c       transfer scale parameters to cubic function calculations
c       so that the curves can be plotted

            ex(k+2)=x(n+1)
            ex(k+3)=x(n+2)
            why(k+2)=y(n+1)
            why(k+3)=y(n+2)

c       plot curve

            call line(ex,why,k+1,1,0,4)

25      continue

        pen=pen+1

c       determine whether speed restriction plot or new line plot is perform

        if(pen.eq.3)then
            goto 73
        elseif(pen.eq.4)then
            goto 83
        endif

c       change pens and move pen to new origin
```

```
      call newpen(2)
      call plot(0.,6.6,-3)

c     Plot rolling resistance axis and calculate scale.

      call myscale(RR,n,minrr,maxrr,6.5,firstrr,factrr)
      RR(n+1)=firstrr
      RR(n+2)=factrr
      call chart(pen,x,RR,x,RR,n,namea,nameb,xaxis,raxis,taxis,yaxis2)
      do 55, i=1,n+2
          y(i)=RR(i)
55    continue
      goto 63

c     change pens and move pen to new origin

73    call newpen(3)
      call plot(0.,6.6,-3)

c     Plot speed restriction axis, top axis and calculate scale.

      call myscale(SR,n,minsr,maxsr,6.5,firstsr,factsr)
      SR(n+1)=firstsr
      SR(n+2)=factsr
      call chart(pen,x,SR,x,SR,n,namea,nameb,xaxis,yaxis3,taxis,raxis)
      do 65, i=1,n+2
          y(i)=SR(i)
65    continue

c     plot speed restriction lines

      call line(x,y,n,1,0,4)

c     repeat procedure for more lines, moving origin to new piece of paper

83    if(pointd.eq.000)then
          call plot(-25.,-10.2,-3)
          goto 333
      endif

c     end of plot

      call plot(16.5,0.0,999)

      stop
      end
c*************************************************************************

      subroutine minmax(x,maxx,minx)

c     subroutine to determine the maximum and minimum values of x

      real x,maxx,minx

c     x=data point
c     maxx=maximum of x
c     minx=minimum of x

      if(maxx.lt.x)then
          maxx=x
      endif

      if(minx.gt.x)then
          minx=x
      endif
```

```
c       return to main program

        return
        end
```

A2 - INPUT/OUTPUT DATA FILE LISTINGS

		Page
1.	Digit Output	50-52
2.	Digcuspdis Output	53-55

```
filename:  in
distance units:  metres
elevation units:  metres
rolling resistance units:  per cent
speed restriction units:  kmh
old powder road
DATA
  1 RR      9.000
  1 EL   1000.000
  1 SR     50.000
  6 EL   1000.000
  7 RR      9.100
 11 EL   1050.000
 14 RR      9.600
 14 EL   1075.000
 19 RR      9.800
 19 EL   1080.000
 24 RR     10.100
 24 EL   1125.000
 28 RR     10.340
 28 EL   1162.000
 28 SR     50.000
000 FF 0.0
DISTANCE
  1   54.39
  2   55.44
  3    9.51
  4    6.39
  5    7.63
  6   40.02
  7   29.70
  8   31.10
  9   15.10
 10   26.67
 11   38.59
 12   37.89
 13   22.82
 14   40.87
 15   23.68
 16   32.13
 17   38.53
 18   25.81
 19   31.06
 20   19.28
 21   25.50
 22   22.47
 23    9.33
 24    8.75
 25   11.11
 26   12.05
 27   33.62
000 0.0
dam haul
DATA
  1 RR      5.000
  1 EL    750.000
  1 SR     30.000
  5 EL    725.000
  6 EL    700.000
  7 EL    695.000
  7 SR     50.000
 10 RR      7.000
 10 EL    745.000
 16 RR      7.700
 16 EL    745.000
 20 RR      9.100
```

```
 20 EL    710.000
 24 RR      9.780
 24 EL    693.000
 24 SR     50.000
000 FF 0.0
DISTANCE
  1   30.02
  2   41.17
  3   22.62
  4   23.04
  5   24.35
  6   30.93
  7   33.38
  8   30.42
  9   24.73
 10   14.42
 11   27.02
 12   33.07
 13   47.08
 14   16.31
 15   22.62
 16   32.10
 17   27.00
 18   22.83
 19   23.67
 20   20.69
 21   17.17
 22   17.88
 23   15.69
000 0.0
ridge-haul
DATA
  1 RR      9.100
  1 EL   1000.000
  1 SR     50.000
  5 RR      9.200
  5 EL   1015.000
 11 RR      9.500
 11 EL   1105.000
 13 EL   1110.000
 19 EL   1111.000
 24 EL   1167.000
 28 EL   1170.000
 33 RR      9.600
 36 RR     10.200
 36 EL   1235.000
 36 SR     50.000
000 FF 0.0
DISTANCE
  1   40.17
  2   39.97
  3   40.83
  4   39.35
  5   37.28
  6   45.48
  7   44.88
  8   37.87
  9   35.49
 10   15.05
 11   26.50
 12   22.47
 13   25.67
 14   31.92
 15   38.22
 16   35.51
 17   30.07
```

18	29.85
19	23.94
20	20.47
21	32.75
22	33.51
23	31.15
24	37.97
25	31.96
26	37.20
27	34.33
28	31.96
29	31.87
30	28.94
31	26.83
32	28.38
33	22.92
34	28.22
35	38.70

```
filename:  in
x-value range:      25.00000      -     500.0000
y-value range:    0.0000000E+00 -     393.3330
north arrow, point 1:      489.5203        380.8707
north arrow, point 2:      501.2357        387.1908
plot units:  metres
elevation units:  metres
rolling resistance units:  per cent
speed restriction units:  kmh

old powder road
A 282.60 191.24
 &  RR       9.000
 &  EL    1000.000
 %  SR      50.000
B 304.67 141.53
B 327.31  90.93
B 332.34  83.03
B 338.07  83.78
C 339.94  90.95
 %  EL    1000.000
C 337.35 130.85
 %  RR       9.100
B 344.10 159.36
B 369.11 177.26
B 383.44 180.42
C 406.11 166.63
 %  EL    1050.000
B 443.28 156.39
B 478.45 143.00
C 486.38 122.58
 &  RR       9.600
 %  EL    1075.000
B 476.67  83.08
B 464.41  63.13
B 436.30  48.01
B 398.62  41.04
C 375.29  50.22
 &  RR       9.800
 %  EL    1080.000
B 364.42  78.83
B 367.33  97.47
B 385.79 114.87
B 402.35 130.02
C 410.56 131.97
 &  RR      10.100
 %  EL    1125.000
B 410.01 124.34
B 406.68 113.87
B 412.07 103.53
D 442.19  88.72
 &  RR      10.340
 &  EL    1162.000
 %  SR      50.000

dam haul
A  32.56 106.93
 &  RR       5.000
 &  EL     750.000
 %  SR      30.000
B  51.22 130.23
B  88.57 146.26
B 106.51 134.54
C 112.74 112.64
 %  EL     725.000
C 132.09  98.61
```

```
 %  EL    700.000
C 162.44  97.60
 &  EL    695.000
 %  SR     50.000
B 187.72 118.86
B 206.50 142.76
C 225.02 158.60
 &  RR      7.000
 %  EL    745.000
B 234.98 151.44
B 232.77 124.69
B 229.49  91.78
B 222.82  45.18
B 219.36  29.58
C 199.74  18.95
 &  RR      7.700
 %  EL    745.000
B 168.14  13.58
B 141.45  10.82
B 122.05  22.17
C  99.74  15.10
 &  RR      9.100
 %  EL    710.000
B  79.57  16.59
B  73.19  31.37
B  83.25  44.02
D  91.14  32.99
 &  RR      9.780
 &  EL    693.000
 %  SR     50.000

ridge-haul
A  33.33 199.82
 &  RR      9.100
 &  EL   1000.000
 %  SR     50.000
B  34.81 239.96
B  37.18 279.84
B  46.70 319.36
C  71.59 349.50
 &  RR      9.200
 %  EL   1015.000
B 103.35 368.74
B 146.57 382.67
B 190.96 388.36
B 227.30 379.51
B 248.67 351.66
C 252.30 337.52
 &  RR      9.500
 %  EL   1105.000
B 239.16 315.86
C 220.52 324.39
 %  EL   1110.000
B 211.30 347.96
B 183.52 362.48
B 145.93 357.80
B 114.91 340.84
B  93.71 319.63
C  77.13 294.86
 %  EL   1111.000
B  68.32 272.66
B  66.05 252.63
B  82.88 225.12
B 114.07 213.77
C 144.92 210.10
 %  EL   1167.000
```

```
B 182.46 214.62
B 208.97 231.66
B 223.19 265.41
C 213.92 297.81
 %   EL    1170.000
B 189.28 317.38
B 158.06 320.88
B 133.25 306.61
B 115.48 286.81
C 109.92 259.67
 %   RR       9.600
B 126.21 245.98
B 151.79 256.11
# 177.12 285.24
 &   RR      10.200
 &   EL    1235.000
 %   SR      50.000
```

A3 - SUBROUTINE LISTINGS

		Page
1.	Scale for File	57
2.	Cugraph for File	58
3.	Grade for File	59
4.	Angle for File	60-62
5.	Transform for File	63-66
6.	Spline for File	67
7.	Segment for File	68
8.	Chart for File	69-70

```
      subroutine myscale(array,n,min,max,length,first,fact)

c     subroutine to calculate scaling factor of an array

      real array(n+2),first,fact,max,min,delta,large,length
      integer n

c     array-array for which scaling factors are calculated
c     first-first value for axis
c     large-adjusted maximum element of array
c     fact-scaling factor
c     max-maximum array element
c     min-minimum array element
c     delta-unadjusted scaling factor
c     n-# of data points in array
c     length-axis length
c     small-adjusted first value used in fact calculation

      first=aint(min)
      small=first-1
      large=anint(max)+1
      delta=(large-small)/length
      fact=anint(delta)+1

c     return to main program

      return
      end

c*************************************************************************

      subroutine charcount(string,n)

c     subroutine to determine length of a character string (count ends
c     when a double blank is encountered)

      integer n
      character*(*) string

c     n-# of characters in string
c     string-character string

c     compare character values

      n=0

      do 10, i=1,len(string)
          if(string(i:i+1).gt.'  ')then
             n=n+1
          endif
10    continue

c     return to main program

      return
      end
```

```
      subroutine cugraph(firstx,factx,firsty,facty,namea,nameb,xaxis,yaxis

c     a program to plot a graph, based on requirements of the digcuspdis
c     program

c     firstx,firsty=first values for each axis
c     factx,facty=scaling factor
c     namea,nameb= title lines of graph
c     z,t=# of characters in title lines
c     xaxis,yaxis=titles of x and y axis respectively
c     w,v=# of characters in x and y axis titles

      real firstx,firsty,factx,facty
      integer z,t,w,v
      character*(*) xaxis, yaxis, namea,nameb

c     determine values for z,t,w,v

      call charcount(namea,z)
      call charcount(nameb,t)
      call charcount(xaxis,w)
      call charcount(yaxis,v)

c     plot the title of the graph

90    call symbol(0.5,21.6,0.21,namea,14,0.0,z)
100   call symbol(0.5,21.2,0.14,nameb,14,0.0,t)

c     plot the axes and the titles of the axes

60    call axis(0.,0.,xaxis,-w,15.5,0.,firstx,factx)
70    call axis(0.0,0.0,yaxis,v,21.0,90.0,firsty,facty)

c     return to the main program

120   return
      end

c*************************************************************************

      subroutine dataplt(x,y,n,sym)

c     subroutine to plot symbols

      real x(n+2),y(n+2)
      integer n,sym

c     x,y=data points
c     n=# of data lines
c     sym=# used to determine what symbol is plotted

c     plot the data points

      call line(x,y,n,1,-1,sym)

c     return to main program

      return
      end
```

```
      subroutine grade(x,y,n,j,s)

c     subroutine to calculate grades (slopes) at certain points by using
c     a slope parallel to a slope between the two adjacent points

c     x,y=data points
c     n=# of data lines in file
c     s=slope

      real x(n),y(n),s(n)
      integer n,j

c     for the slope of the first point

      if(j.eq.1)then

c         if slope is infinity, assign a large value

          if((x(j+1)-x(j)).eq.0)then
              s(j)=999999.999
              goto 73
          endif

          s(j)=((y(j+1)-y(j))/(x(j+1)-x(j)))

73        if((x(j+2)-x(j)).eq.0)then
              s(j+1)=999999.999
              goto 83
          endif

          s(j+1)=((y(j+2)-y(j))/(x(j+2)-x(j)))

c     for the slope of the last point

      elseif(j.eq.(n-1))then

c         if slope is infinity, assign a large value

          if((x(j+1)-x(j)).eq.0)then
              s(j+1)=999999.999
              goto 83
          endif

          s(j+1)=((y(j+1)-y(j))/(x(j+1)-x(j)))

c     for the slope of all other points

      else

c         if slope is infinity, assign a large value

          if((x(j+2)-x(j)).eq.0)then
              s(j+1)=999999.999
              goto 83
          endif

          s(j+1)=((y(j+2)-y(j))/(x(j+2)-x(j)))

83    endif


c     return to main program

      return
      end
```

```
c*******************************************************************************

      subroutine angle(x,y,n,j,s)

c     subroutine to calculate the slope at each point by using angles

c     x,y=data points
c     n=# of data lines
c     s=slope at point
c     alpha=angle of slope at point
c     sigma=angle between slope angle and hypotenuse formed by point
c           and point agter
c     rho=angle formed between hypotenuse and x-axis
c     tau=angle formed between hypotenuse and y-axis
c     d=distance between two points (hypotenuse)

      real x(n),y(n),s(n),d1,d2,alpha,rho,tau,pi,mu,chi
      integer n,j

      pi=3.1414927

c     for the first and last slopes, a different calculation is made
c     based on delta y over delta x

c     for the slope at the first point

      if(j.eq.1)then
          if((x(j+1)-x(j)).eq.0)then
              s(j)=999999.999
          else
              s(j)=(y(j+1)-y(j))/(x(j+1)-x(j))
          endif
          goto 100

c     for the slope at the last point

      elseif(j.eq.n)then
          if((x(j)-x(j-1)).eq.0)then
              s(j)=999999.999
          else
              s(j)=(y(j)-y(j-1))/(x(j)-x(j-1))
          endif
          goto 100
      endif

c     calculate the distance between points

      d1=sqrt(((y(j)-y(j-1))**2)+((x(j)-x(j-1))**2))
      d2=sqrt(((y(j+1)-y(j))**2)+((x(j+1)-x(j))**2))

c     calculate the angles needed to calculate the slope

      tau=atan((x(j)-x(j-1))/(y(j)-y(j-1)))
      mu=atan((x(j+1)-x(j))/(y(j+1)-y(j)))

c     if the distance between 1 and 2, and between 2 and 3 are
c     the same, the following conditions apply

      if(d2.eq.d1)then
          if((abs(tau).eq.abs(mu)).and.(x(j-1).eq.x(j+1)))then
              s(j)=999999.999
          elseif((abs(tau).eq.abs(mu)).and.(y(j-1).eq.y(j+1)))then
              s(j)=0.00000
          else
              d2=d2+1
              write(2,*),'d2=d1'
```

```
                goto 799
            endif
            goto 100
799     endif

c       rho must have the same sign as tau for the sign of the slope to
c       work out correctly, and chi is assigned an appropiate value (tau-pi)

        if(tau.lt.0)then
            rho=-(pi/2.)-tau
            chi=-rho
        else
            rho=(pi/2.)-tau
            chi=-rho
        endif

c       calculation of alpha (angle corresponding to the slope)
c       for each case tau must be considered for four different
c       angles; 1) <pi/8
c               2) pi/8<tau<pi/4
c               3) pi/4<tau<3pi/8
c               4) tau>3pi/8

c       case1:  y1>y2<y3

1       if((y(j-1).gt.y(j)).and.(y(j+1).gt.y(j)))then
            alpha=(rho*d2)/(abs(d2+d1))

c       case2:  y1<y2>y3

2       elseif((y(j-1).lt.y(j)).and.(y(j+1).lt.y(j)))then
            alpha=(rho*d2)/(d1+d2)

c       case3a:  x1>x2>x3 and y1>y2>y3

31      elseif(((x(j-1).gt.x(j)).and.(x(j).gt.x(j+1))).and.
3           ((y(j-1).gt.y(j)).and.(y(j).gt.y(j+1))))then
            if(abs(tau).lt.(pi/8.))then
                alpha=(chi*d2)/(abs(d2-d1))
            else
                alpha=(rho*d2)/(d2+d1)
            endif

c       case3b:  x1>x2>x3 and y1<y2<y3

32      elseif(((x(j-1).gt.x(j)).and.(x(j).gt.x(j+1))).and.
3           ((y(j-1).lt.y(j)).and.(y(j).lt.y(j+1))))then
            IF(ABS(TAU).LT.(PI/8.))THEN
                ALPHA=(chi*D2)/(ABS(D2-D1))
            ELSE
                alpha=(rho*d2)/(d2+d1)
            endif

c       case4:  x1<x2>x3

4       elseif((x(j-1).lt.x(j)).and.(x(j+1).lt.x(j)))then
            IF(ABS(TAU).LT.(PI/8.))THEN
                ALPHA=(rho*d2)/(abs(d2-d1))
            ELSEIF(ABS(TAU).LT.(PI/4.))THEN
                ALPHA=(chi*D2)/(ABS(D2-D1))
            ELSE
                alpha=(rho*d2)/(abs(d2+d1))
            endif

c       case5:  x1>x2<x3
```

```
5       elseif((x(j-1).gt.x(j)).and.(x(j+1).gt.x(j)))then
            if(abs(tau).lt.(pi/4.))then
                alpha=(chi*d2)/(abs(d2-d1))
            elseif(abs(tau).lt.((3./8.)*pi))then
                alpha=(rho*d2)/(abs(d2-d1))
            else
                alpha=(rho*d2)/(abs(d2+d1))
            endif

c       case6:  x1<x2<x3 and y1<y2<y3

6       elseif(((x(j-1).lt.x(j)).and.(x(j).lt.x(j+1))).and.
        3       ((y(j-1).lt.y(j)).and.(y(j).lt.y(j+1))))then
            if(abs(tau).lt.(pi/8.))then
                alpha=(rho*d2)/(abs(d2-d1))
            else
                alpha=(rho*d2)/(d2+d1)
            endif

c       case7:   x1<x2<x3 and y1>y2>y3

7       elseif(((x(j-1).lt.x(j)).and.(x(j).lt.x(j+1))).and.
        3       ((y(j-1).gt.y(j)).and.(y(j).gt.y(j+1))))then
            if(abs(tau).lt.(pi/8.))then
                alpha=(rho*d2)/(ABS(d2-d1))
            else
                alpha=(rho*d2)/(abs(d2+d1))
            endif
        endif

c       calculate slope

        s(j)=tan(alpha)

c       return to main program

100     return
        end
```

```
      subroutine transform(p,x,y,j,e,q,r,n,theta,tx,ty,nq,nr)

c     subroutine to transform co-ordinates when an angle of rotation
c     is to be determined by a second point or is given

c     p=point transform.le.0 (theta not given)
c     p=rotation transform, p>0 (theta given)
c     h=hypothenus
c     theta=axis rotation, in radians
c     beta=hypothenus rotation, in radians
c     gamma=total rotation, in radians
c     q=x(j)=x offset
c     nq=x-offset in terms of new origin
c     r=y(j)=y offset
c     nr=y offset in terms of new origin
c     e=# times calculation is performed (=0 if axis rotation is to be
c           calculated)
c     tx,ty=new co-ordinates
c     n=# of lines in data file

      real x(n),y(n),h,theta,beta,gamma,tx(n),ty(n),q,r,nq,nr
      integer j,p,e,n

c     calculate origin and end point of line

      if(p.le.0)then
          tx(1)=x(j)-q
          ty(1)=y(j)-r
          tx(e+2)=sqrt(((x(j+1)-x(j))**2)+((y(j+1)-y(j))**2))
          ty(e+2)=0.0
      endif

c     calculate theta

      if(p.le.0)then
          call thetaa(x(j+1),q,y(j+1),r,theta)
      endif

c     transform points other than beginning point of segment

      if(e.eq.0)then
          goto 77

c     calculation of angle between original axis and hypotenuse (beta)

      else
          do 20, i=1,e
              h=sqrt(((x(i)-q)**2)+((y(i)-r)**2))
              call betaa(x(i),q,y(i),r,beta)

c     determine the total rotation of the line segment

              call gammaa(x(i),q,beta,theta,gamma)

c     calculate the new x,y co-ordinates

              tx(i)=h*(cos(gamma))
              ty(i)=h*(sin(gamma))

20        continue

c     calculate the new q and r for use if points will later be
c     transformed back to original co-ordinates

c     if q and r = 0, then they do not need to be transformed
```

```
77          if((q.eq.0).and.(r.eq.0))then
                nq=q
                nr=r
                goto 83
            endif

c       transform q and r to the  new co-ords

            h=sqrt(((0.0-q)**2)+((0.0-r)**2))
            call  betaa(0.0,q,0.0,r,beta)
            call gammaa(0.0,q,beta,theta,gamma)
            nq=h*(cos(gamma))
            nr=h*(sin(gamma))
83      endif

c       return to main program

        return
        end

c*******************************************************************************

        subroutine thetaa(x,q,y,r,theta)

c       subroutine to calculate the angle between the original axis and the
c       new axis

c       x,y=data points
c       q,r=x and y offsets, respectively
c       theta= angle of axis rotation, in radians

        real x,q,y,r,theta

        if(((x-q).eq.0).and.((y-r).gt.0))then
            theta=1.5707963
        elseif(((x-q).eq.0).and.((y-r).lt.0))then
            theta=-1.5707963
        elseif((x-q).lt.0)then
            theta=(atan((y-r)/(x-q)))-3.1415927
        else
            theta=atan((y-r)/(x-q))
        endif

c       return to main program

        return
        end

c*******************************************************************************

        subroutine betaa(x,q,y,r,beta)

c       subroutine to determine the angle between the hypotenuse and the
c       x-axis

c       x,y=data points
c       q,r=x and y offsets respectively
c       beta=rotation of line segment from original axis, in radians

        real x,y,q,r,beta

        if(((x-q).eq.0).and.((y-r).gt.0))then
            beta=1.570963
        elseif(((x-q).eq.0).and.((y-r).lt.0.0))then
            beta=-1.5707963
        elseif(((x-q).eq.0).and.((y-r).eq.0))then
```

```
            beta=0.0
        else
            beta=atan((y-r)/(x-q))
        endif

c       return to main program

        return
        end

c*******************************************************************************

        subroutine gammaa(x,q,beta,theta,gamma)

c       subroutine to determine the total rotation of the line segment from
c       the new axis

c       x=data point
c       q=x offset
c       beta= angle of rotation of segment from original axis, in radians
c       theta=angle of rotation of new axis from original axis, in radians
c       gamma=angle of rotation of segment from new axis, in radians

        real x,q,beta,theta,gamma

        if(x.lt.q)then
            gamma=(beta-theta)-3.1415927
        else
            gamma=beta-theta
        endif

c       return to main program

        return
        end

c*******************************************************************************

        subroutine angslope(theta,s1,s2,ns1,ns2)

c       subroutine to change a slope to an angle, convert the angle to new
c       co-ordinates, then change it back to a slope

c       theta=axis rotation, in radians
c       s1,s2=original slopes
c       ns1,ns2=transformed slopes
c       angle1,angle2=original angle of original slopes, in radians
c       newan1,newan2=new angle of original slopes,in radians

        real theta,s1,s2,ns1,ns2,newan1,newan2,angle1,angle2

c       calculate original angles

        angle1=atan(s1)
        angle2=atan(s2)

c       calculate the new angles

        newan1=angle1-theta
        newan2=angle2-theta

c       calculate the new slopes

        ns1=tan(newan1)
        ns2=tan(newan2)
```

```
c       return to main program

77      return
        end
```

```
      subroutine spline(x1,x2,y1,y2,s1,s2,a,b,c,d)
c     a subroutine to fit a cubic function between 2 points
      real x1,x2,y1,y2,s1,s2,a,b,c,d,num,denom
c     cubic function equations
c     y1=a+b*x1+c*(x1**2)+d*(x1**3)
c     y2=a+b*x2+c*(x2**2)+d*(x2**3)
c     slope function equations are the differential of the cubic
c     equations
c     s1=b+2*c*x1+3*d*(x1**2)
c     s2=b+2*c*x2+3*d*(x2**2)
c     mathematical manipulation produces the following equations
c     to solve for a,b,c,d
      num=(y1-y2)-(s2*(x1-x2))+(x2*(s1-s2))-(0.5*(x1+x2)*(s1-s2))
      denom=(0.5*(x2**3))-(0.5*(x1**3))+((3./2.)*(x1**2)*x2)-((1.5)
     3       *x1*(x2**2))
c     if the denominator = 0 then d is assumed to be 0
      if(denom.eq.0)then
          d=0.
      else
          d=num/denom
      endif
c     if x1=x2 then c is 0
      if((x1-x2).eq.0)then
          c=0.
      else
          c=((s1-s2)-(3*d*((x1**2)-(x2**2))))/(2*(x1-x2))
      endif
      b=s1-(2*c*x1)-(3*d*(x1**2))
      a=y1-(b*x1)-(c*(x1**2))-(d*(x1**3))
c     return to main program
      return
      end
```

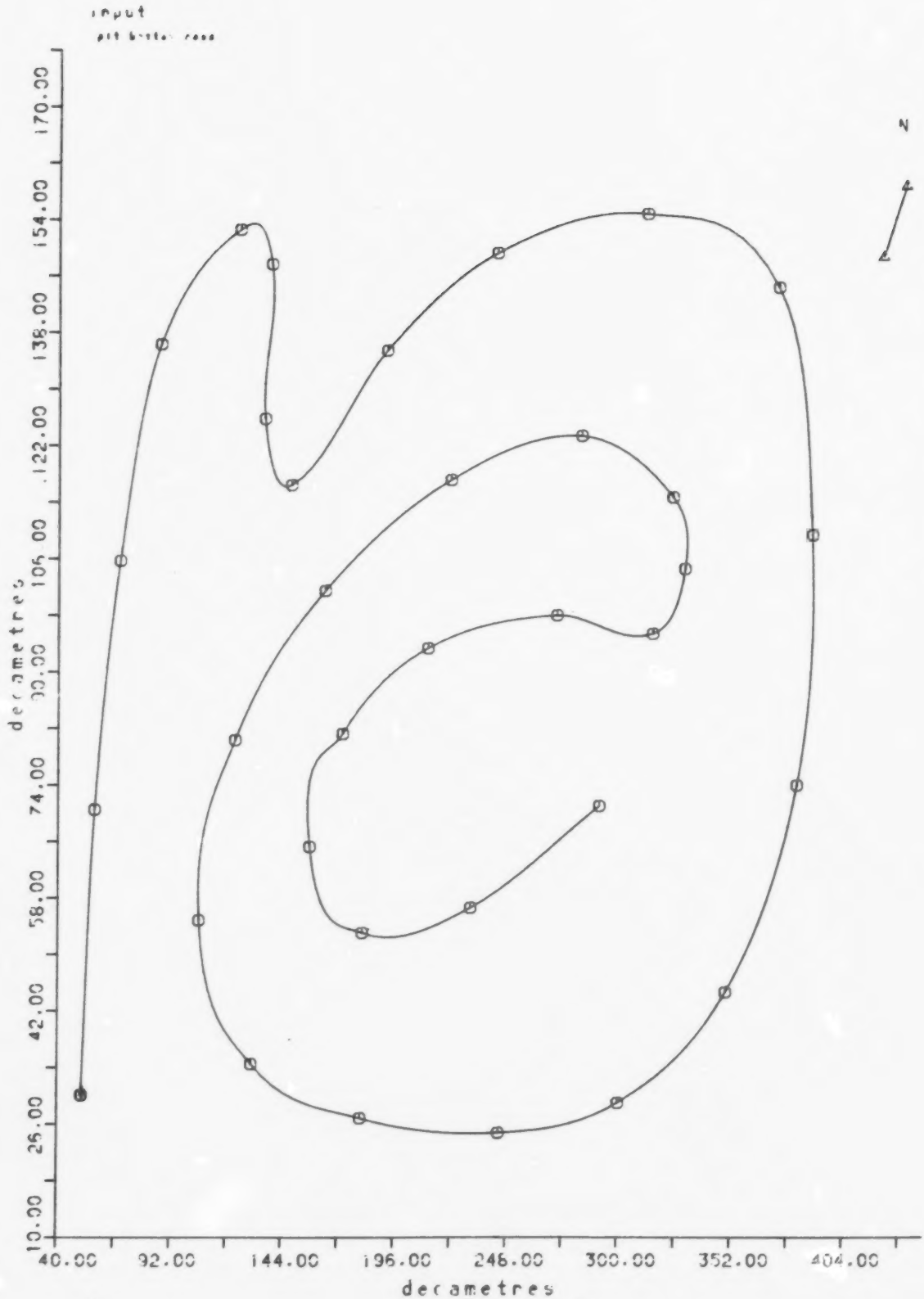
```
      subroutine segment(x,y,j,n,k)

c     subroutine that determines the number of segments that the spline
c     function will split the line, between two given points, into

c     x,y=data points
c     k=# of segments the line between two points is divided into
c     n=# of data lines in file

      real x(n),y(n)
      integer n,j,k

c     the number of segments depends on the length of the line

      if((abs(x(j+1)-x(j)).gt.25).or.(abs(y(j+1)-y(j)).gt.25))then
          k=50
      elseif((abs(x(j+1)-x(j)).gt.5).or.(abs(y(j+1)-y(j)).gt.5))then
          k=25
      elseif((abs(x(j+1)-x(j)).gt.1).or.(abs(y(j+1)-y(j)).gt.1))then
          k=15
      else
          k=10
      endif

c     return to main program

      return
      end
```

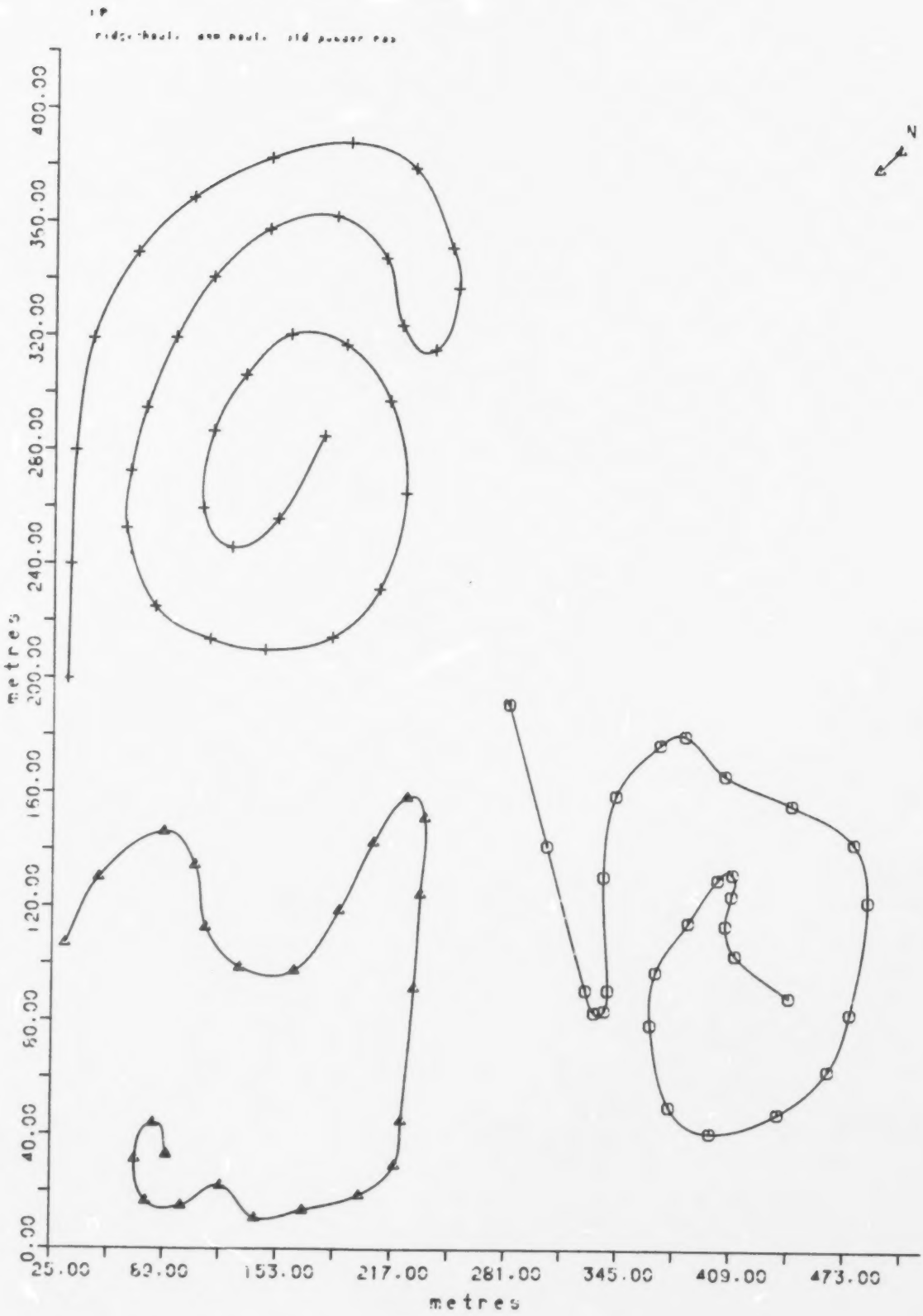
```
      subroutine chart(pen,x,y,top,r,n,namea,nameb,xaxis,
     3       yaxis,topaxis,raxis)

c     a  subroutine to plot a four sided chart ( or graph), based on the
c     requirements of the cross program

      real x(n+2),y(n+2),top(n+2),r(n+2)
      integer n,z,t,w,v,m,f,pen
      character*30 xaxis,yaxis,topaxis,raxis,namea,nameb

c     x,y=x,y data points
c     top,r=axis at top and right-hand side of page, with
c           corresponding data points
c     z,t=#of characters in title lines
c     n=# of data lines in file
c     w,v,m,f=# of characters in axis titles;  x,y,top,r, respectively
c     xaxis,yaxis,taxis,baxis = titles of each axis
c     namea,nameb=title of chart
c     pen=# of call to chart

c     determine values for z,t,w,v,m,f

      if(pen.eq.1)then
          call charcount(namea,z)
          call charcount(nameb,t)
          call charcount(xaxis,w)
          call charcount(yaxis,v)
          call charcount(raxis,f)
      elseif(pen.eq.3)then
          call charcount(yaxis,v)
          call charcount(raxis,f)
          call charcount(topaxis,m)
      else
          call charcount(yaxis,v)
          call charcount(raxis,f)
      endif

c     for plotting of first part of graph

      if(pen.eq.1)then

c     initialize and open the plotting routine

          call plots(0,0,6)

c     create a  new origin

          call plot(0.,0.,-3)

c     plot the title of the graph

          call symbol(0.5,20.8,0.21,namea,14,0.,z)
          call symbol(0.5,20.4,0.14,nameb,14,0.,t)

c     plot the x-axis and its title

          call axis(0.,0.,xaxis,-w,15.5,0.,x(n+1),x(n+2))

c     plot remaining axis and its title

      elseif(pen.eq.3)then
          call axis(0.,6.5,topaxis,m,15.5,0.,top(n+1),top(n+2))
      endif

c     plot y and r axis (common to all calls to chart)
```

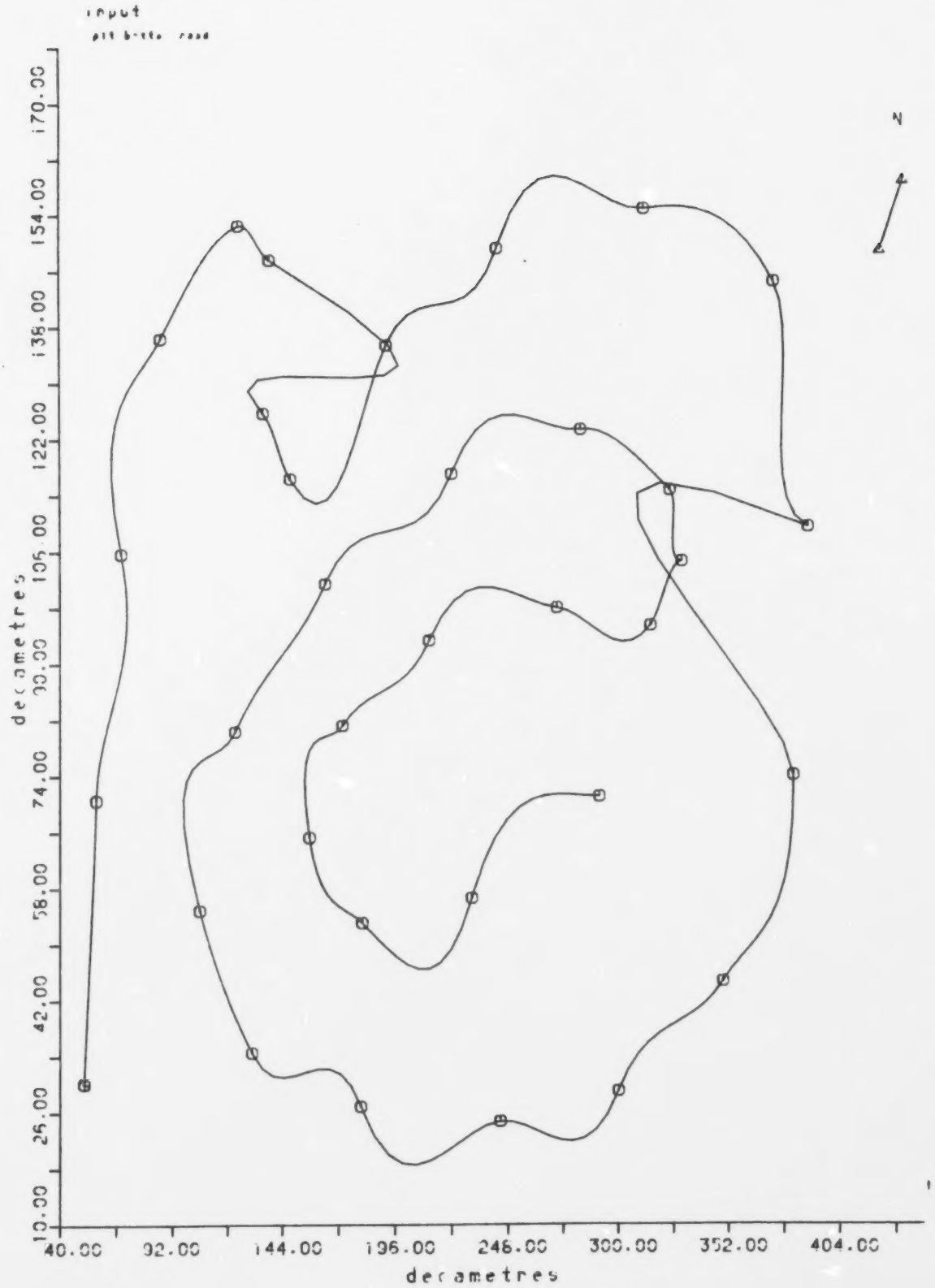
```
      if((pen.eq.1).or.(pen.eq.3))then
          call axis(0.,0.,yaxis,v,6.5,90.,y(n+1),y(n+2))
          call axis(15.5,0.,raxis,f,6.5,90.,r(n+1),r(n+2))
      else
          call axis(0.,0.,yaxis,-v,6.5,90.,y(n+1),y(n+2))
          call axis(15.5,0.,raxis,-f,6.5,90.,r(n+1),r(n+2))
      endif

c     move the pen to the origin

      call plot(0.,0.,3)

c     return to the main program

      return
      end
```

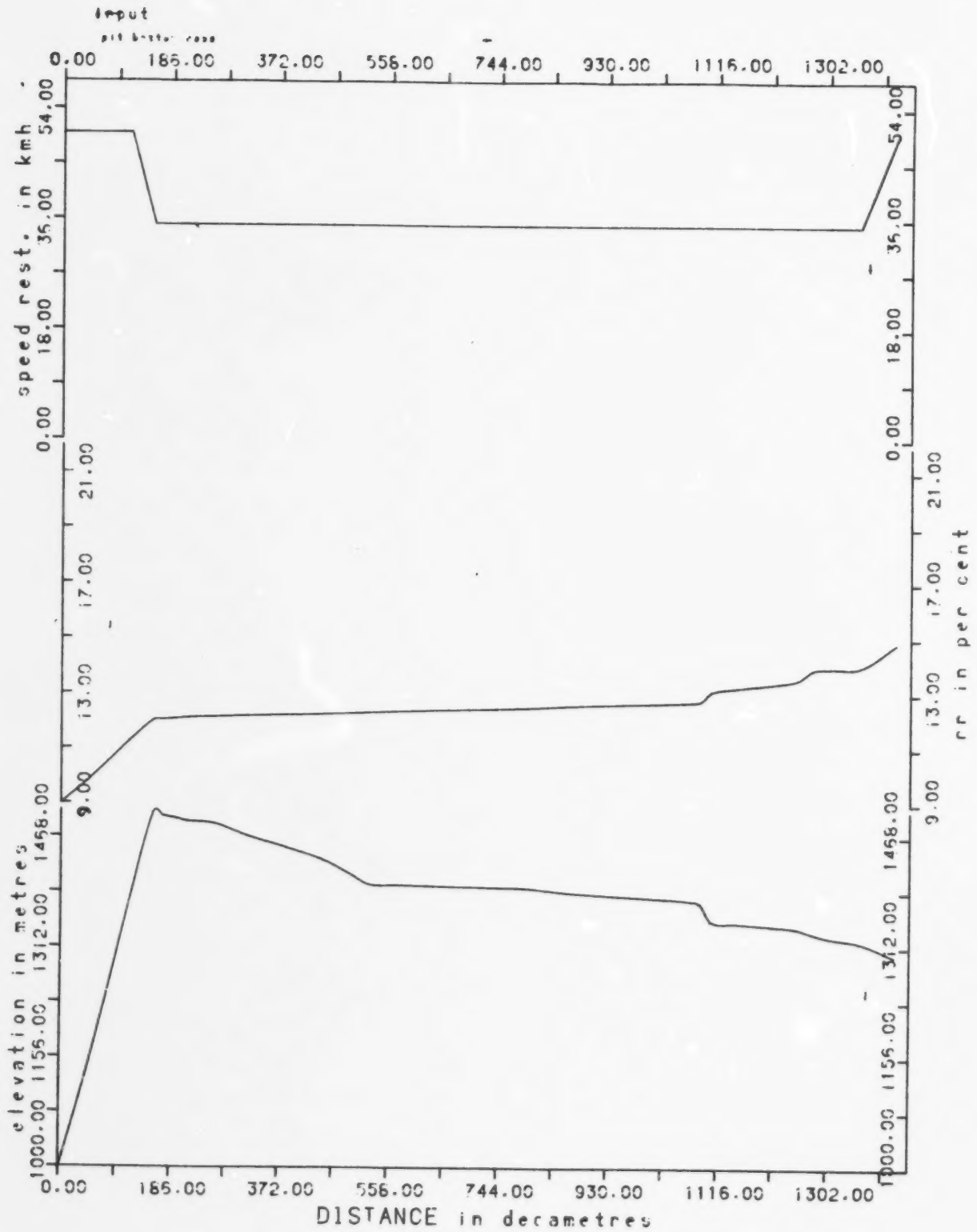
A4 - Graphs

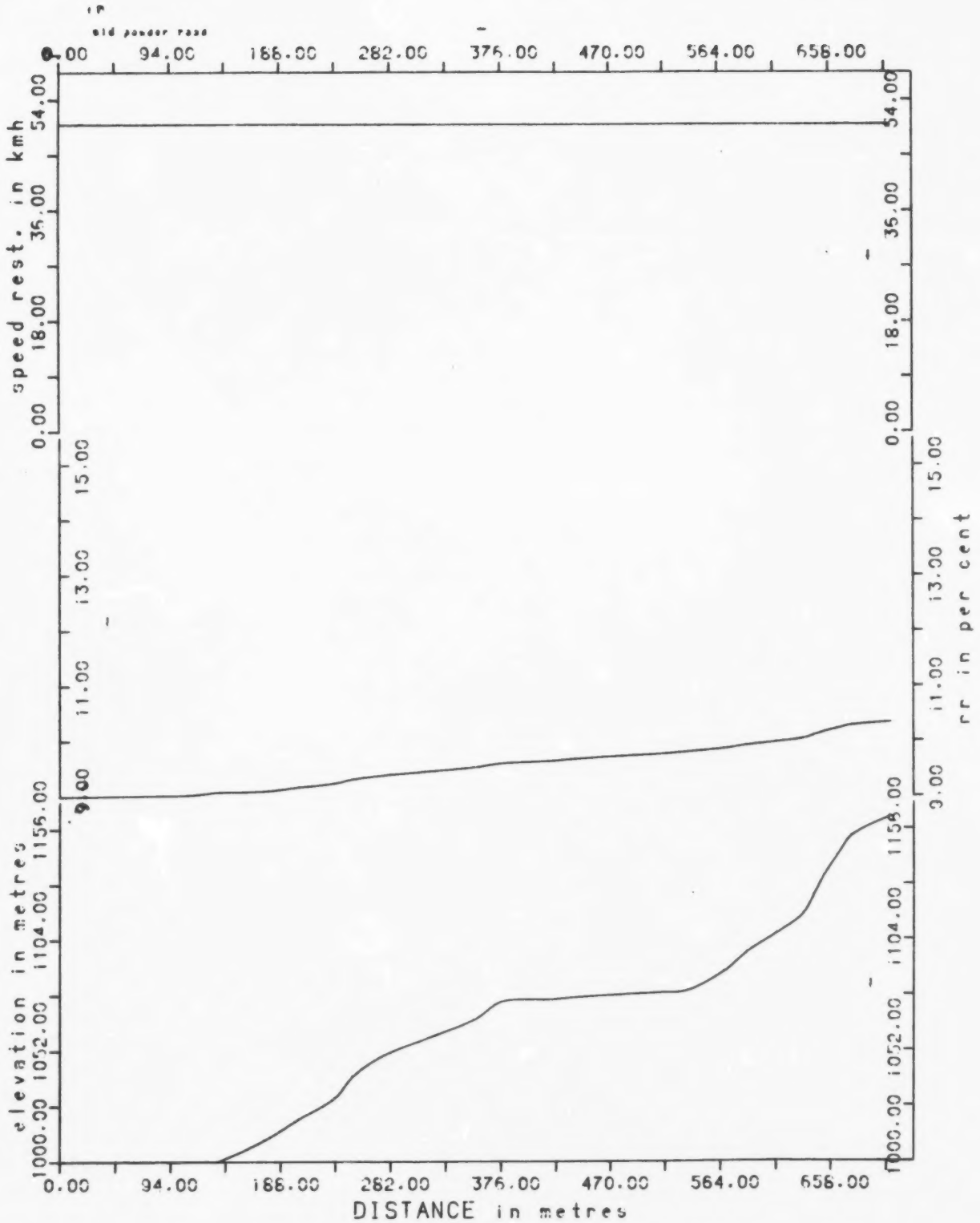
		Page
1.	Digcuspdis Graphs	72-74
2.	Gross Graphs	75-78

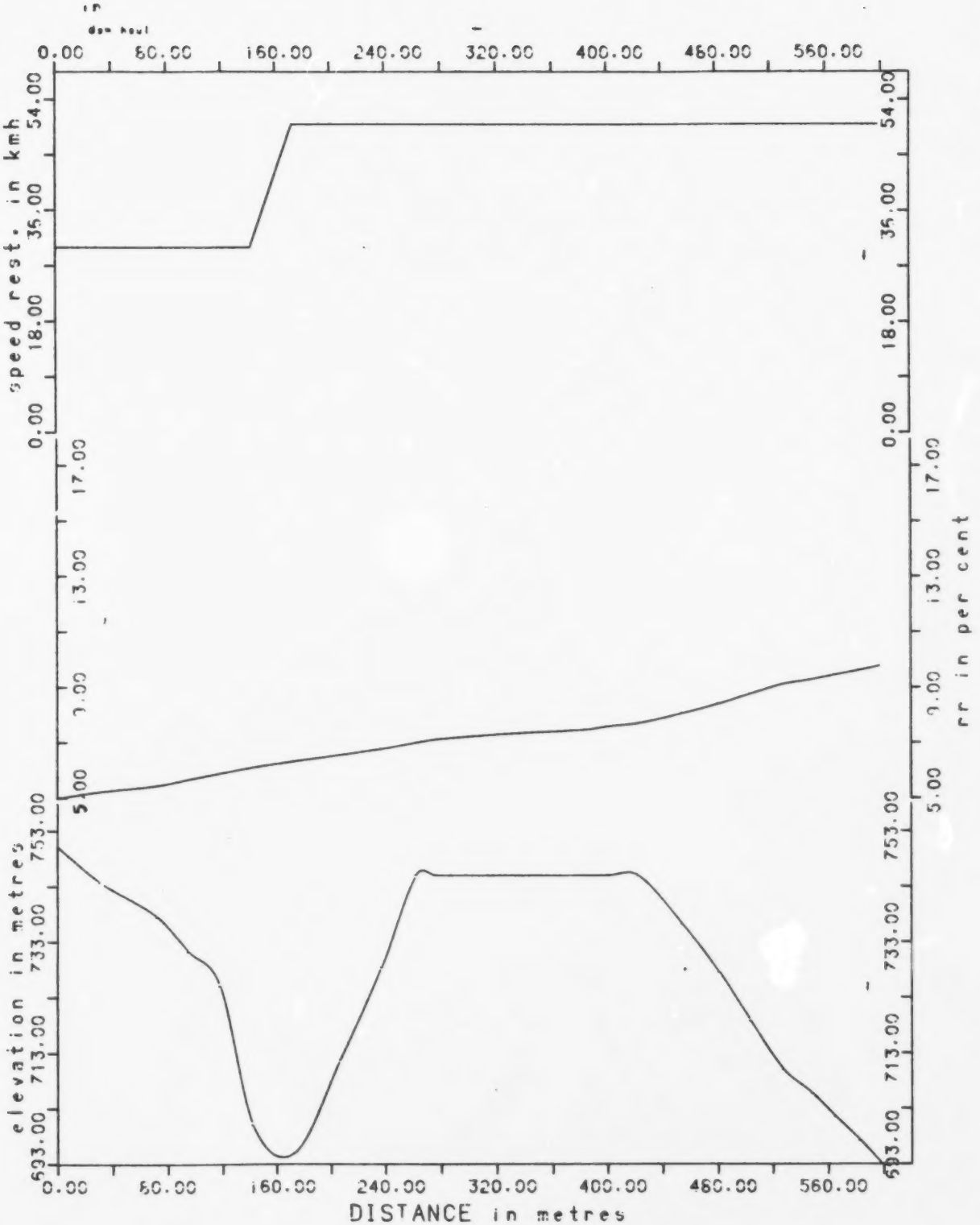

input
pit bottom
N
10.00
26.00
42.00
58.00
74.00
90.00
105.00
122.00
138.00
154.00
170.00
decametres
40.00
92.00
144.00
196.00
248.00
300.00
352.00
404.00
decametres

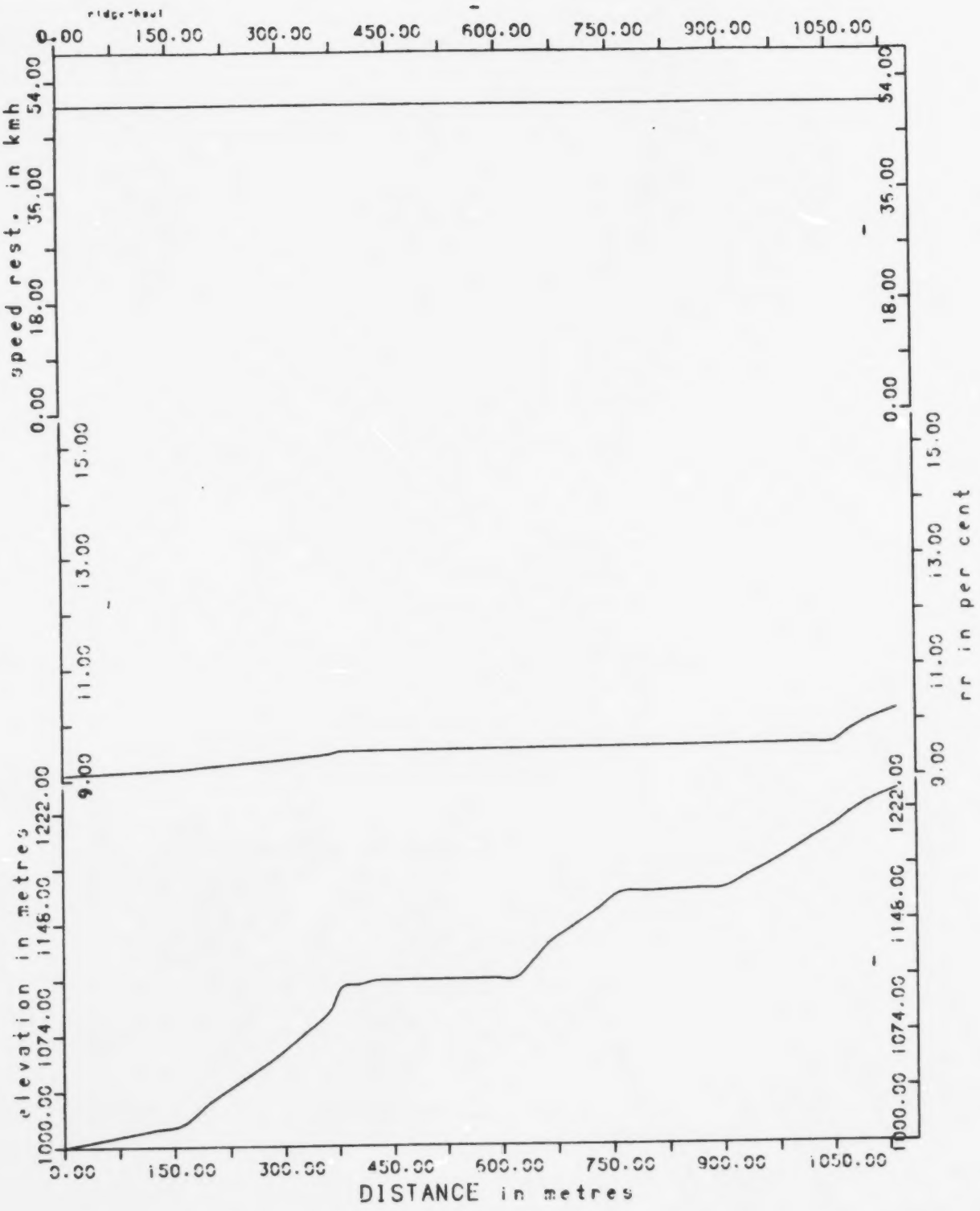

N
0.00
40.00
80.00
120.00
160.00
200.00
240.00
280.00
320.00
350.00
400.00
metres
25.00
89.00
153.00
217.00
281.00
345.00
409.00
473.00
metres


input
pit bottle road
N
decametres
10.00
26.00
42.00
58.00
74.00
90.00
106.00
122.00
138.00
154.00
170.00
40.00
92.00
144.00
196.00
248.00
300.00
352.00
404.00
decametres


Input
0.00
186.00
372.00
556.00
744.00
930.00
1116.00
1302.00
speed rest. in kmh
54.00
36.00
18.00
0.00
21.00
17.00
13.00
9.00
rr in per cent
elevation in metres
1468.00
1312.00
1156.00
1000.00
DISTANCE in decametres


old power pass
0.00
94.00
188.00
282.00
376.00
470.00
564.00
658.00
speed rest. in kmh
0.00
18.00
35.00
54.00
0.00
3.00
11.00
13.00
15.00
rr in per cent
elevation in metres
1000.00
1052.00
1104.00
1156.00
DISTANCE in metres


0.00
80.00
160.00
240.00
320.00
400.00
480.00
560.00
speed rest. in kmh
54.00
35.00
18.00
0.00
17.00
13.00
9.00
5.00
rr in per cent
elevation in metres
753.00
733.00
713.00
693.00
DISTANCE in metres


0.00
150.00
300.00
450.00
600.00
750.00
900.00
1050.00
speed rest. in kmh
0.00
18.00
35.00
54.00
9.00
11.00
13.00
15.00
rr in per cent
elevation in metres
1000.00
1074.00
1146.00
1222.00
DISTANCE in metres